When does data augmentation help generalization in NLP?

Rohan Jha and **Charles Lovering** and **Ellie Pavlick**
Brown University
{first}_{last}@brown.edu

Abstract

Neural models often exploit superficial ("weak") features to achieve good performance, rather than deriving the more general ("strong") features that we'd prefer a model to use. Overcoming this tendency is a central challenge in areas such as representation learning and ML fairness. Recent work has proposed using data augmentation–that is, generating training examples on which these weak features fail–as a means of encouraging models to prefer the stronger features. We design a series of toy learning problems to investigate the conditions under which such data augmentation is helpful. We show that augmenting with training examples on which the weak feature fails ("counterexamples") *does* succeed in preventing the model from relying on the weak feature, but often *does not* succeed in encouraging the model to use the stronger feature in general. We also find in many cases that the number of counterexamples needed to reach a given error rate is independent of the amount of training data, and that this type of data augmentation becomes less effective as the target strong feature becomes harder to learn.

1 Introduction

Neural models often perform well on tasks by using superficial ("weak") features, rather than the more general ("strong") features that we'd prefer them to use. Recent studies expose this tendency by highlighting how models fail when evaluated on targeted challenge sets consisting of "counterexamples" where the weak feature begets an incorrect response. For example, in visual question answering, models failed when tested on rare color descriptions (*"green bananas"*) (Agrawal et al., 2018); in coreference, models failed when tested on infrequent profession-gender pairings (*"the nurse cared for his patients"*) (Rudinger et al., 2018); in natural language inference (NLI), models failed on sentence pairs with high lexical overlap with different meanings (*"man bites dog"*/*"dog bites man"*) (McCoy et al., 2019).

One proposed solution has been to augment training data to over-represent these tail events, often with automatic or semi-automatic methods for generating such counterexamples at a large scale. This technique has been discussed for POS tagging (Elkahky et al., 2018), NLI (McCoy et al., 2019), and as a means of reducing gender bias (Zhao et al., 2018; Zmigrod et al., 2019), all with positive initial results. However, it is difficult to know whether this strategy is a feasible way of improving systems in general, beyond the specific phenomena targeted by the data augmentation. Understanding the conditions under which adding such training examples leads a model to switch from using weaker features to stronger features is important for both practical and theoretical work in NLP.

We design a set of toy learning problems to explore when (or whether) the above-described type of data augmentation helps models learn stronger features. We consider a simple neural classifier in a typical NLP task setting where: 1) the labeled input data exhibits weak features that correlate with the label and 2) the model is trained end-to-end and thus adopts whichever feature representation performs best. Our research questions are:

- How many counterexamples must be seen in training to prevent the model from adopting a given weak feature? Do larger training sets require more counterexamples or fewer?
- Does the relative difficulty of representing a feature (strong or weak) impact when/whether a model adopts it?
- How does the effectiveness of data augmentation change in settings which contain many

weak features but only a single strong feature?

Terminology: This work relates to a very large body of work on learning, generalization, and robustness in neural networks. Our goal with the current set of experiments is to establish a framework within which we can begin to isolate the phenomena of interest (data augmentation in NLP) and observe patterns empirically, without the confounds that exist in the applied settings where the data augmentation techniques in which we are interested have been previously used. The results presented are intended to probe the problem setting and help focus on interesting questions, prior to beginning to formalize the phenomena. As such, our choice of terminology (*"strong"*, *"weak"*, *"hard"*, *"counterexample"*) is meant to be informal. Much of the existing relevant vocabulary carries connotations which we do not specifically intend to invoke here, at least not yet. E.g. the work is related to *adversarial* examples and attacks, but we do not assume any sort of iterative or model-aware component to the way the counterexamples are generated. It is related to *"spurious correlations"*, *"heurisitics"*, and *"counterfactual"* examples, but we do not (yet) make formal assumptions about the nature of the underlying causal graph. We also view the problems discussed as related to work on *perturbations* of the training distribution, to cross-domain generalization, and to non-iid training data; there are likely many options for casting our problem into these terms, and we have not yet determined the best way for doing so. Again, our present goal is to home in on the phenomena of interest and share our initial informal findings, with the hope that doing so will enable more formal insights to follow.

2 Experimental Setup

2.1 Intuition

Our study is motivated by two empirical findings presented in McCoy et al. (2019). Specifically, McCoy et al. (2019) focused on models' use of syntactic heuristics in the context of the natural language inference (NLI) task: given a pair of sentences–the premise p and the hypothesis h–predict whether or not p entails h. They showed that when 1% of the 300K sentence pairs seen in training exhibit lexical overlap (i.e. every word in h appears in p) and 90% of lexical-overlap sentence pairs have the label ENTAILMENT, the model adopts the (incorrect) heuristic that lexical overlap always corresponds to ENTAILMENT. However, after augmenting the training data with automatically generated training examples so that 10% of the 300K training pairs exhibit lexical overlap and 50% of lexical-overlap sentence pairs have the label ENTAILMENT, the same model did *not* adopt the heuristic and appeared to learn features which generalized to an out-of-domain test distribution.

From these results, it is hard to say which changes to the training setup were most important for the model's improved generalizability. The number of lexical-overlap examples seen in training? The probability of ENTAILMENT given that a pair exhibits lexical overlap? Or some other positive artifact of the additional training examples? Thus, we abstract away from the specifics of the NLI task in order to consider a simplified setting that captures the same intuition but allows us to answer such questions more precisely.

2.2 Assumptions and Terminology

We consider a binary sequence classification task. We assume there exists some feature which directly determines the correct label, but which is non-trivial to extract given the raw input. (In NLI, ideally, such a feature is whether the semantic meaning of h contains that of p). We refer to this feature as the **strong feature**. Additionally, we assume the input contains one or more **weak features** which are easy for the model to extract from the input. (This is analogous to lexical overlap between p and h). In our set up, the correct label is 1 if and only if the strong feature holds.[1] However, the strong and weak features frequently co-occur in training and so a model which only represents the weak features will be able to make correct predictions much of the time. We can vary their co-occurence rate by adding **counterexamples** to the training data in which either the strong feature or the weak feature is present, but not both. This setup is shown in Figure 1.

2.3 Implementation

Task. We use a synthetic sentence classification task with sequences of numbers as input and binary $\{0, 1\}$ labels as output. We use a symbolic vocabulary V consisting of the integers $0 \ldots |V|$. In all experiments, we use sequences of length 5 and set $|V|$ to be 50K. We do see some effects asso-

[1] See Appendix B.1 for results which assume varying levels of label noise.

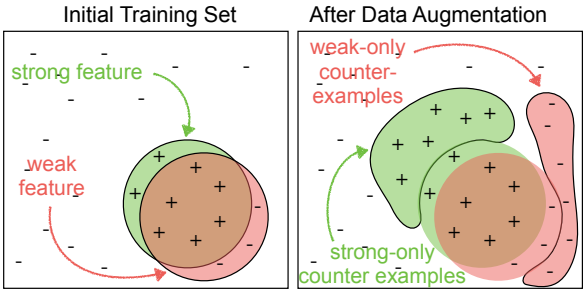

Figure 1: Schematic of experimental setup.

ciated with vocabulary size, but none that affect our primary conclusions; see Appendix A.1 for details.

Model. We use a simple network comprising an embedding layer, a 1-layer LSTM, and a 1-layer MLP with a RELU activation. We found enough interesting trends to analyze in the behavior of this model, and thus leave experiments with more complex model architectures for future work. Our code, implemented in PyTorch, is released for reproducibility.[2] Appendix A discusses how our experimental results extend across changes in model and task parameters. All models are trained until convergence, using early-stopping.[3]

Strong and Weak Features. In all experiments, we set the weak feature to be the presence of the symbol `2` anywhere in the input. We consider several different strong features, listed in Table 1. These features are chosen with the intent of varying how difficult the strong feature is to detect given the raw sequential input. In all experiments, we design train and test splits such that the symbols which are used to instantiate the strong feature during training are never used to instantiate the strong feature during testing. For example, for experiments using `adjacent duplicate`, if the model sees the string `1 4 3 3 15` at test time, we enforce that it never saw any string with the duplicate `3 3` during training. This is to ensure that we are measuring whether the model learned the desired pattern, and did not simply memorize bigrams.

To quantify the difficulty of representing each strong feature, we train the model for the task of predicting directly whether or not the feature holds for each of our candidate feature, using a set of 200K training examples evenly split between cases when the feature does and does not hold. For each feature, Figure 2 shows the validation loss curve (averaged over three runs), flat-lined at its minimum (i.e. its early stopping point). We see the desired gradation in which some feature require significantly more training to learn than others. As a heuristic measure of "hardness", we use the approximate area under this flat-lined loss curve (AUC), computed by taking the sum of the errors across all epochs. Table 1 contains the result for each feature. Note that the weak feature (whether the sequence contains `2`) is exactly as hard as `contains 1`.

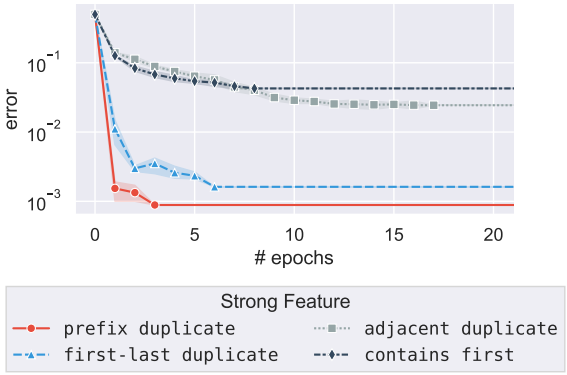

Figure 2: Learning curve for classifying whether the features hold. Averaged over three re-runs and then flat-lined after the average reaches its minimum. Train size is 200K. The error for `contains 1` and the weak feature (not shown) reaches 0 after the first epoch.

2.4 Error Metrics

Definition. We partition test error into four regions of interest, defined below. We use a finer-grained partition than standard true positive rate and false positive rate because we are particularly interested in measuring error rate in relation to the presence or absence of the weak feature.

***weak-only*:**	$P(pred = 1 \mid weak, \neg strong)$
***strong-only*:**	$P(pred = 0 \mid \neg weak, strong)$
***both*:**	$P(pred = 0 \mid weak, strong)$
***neither*:**	$P(pred = 1 \mid \neg weak, \neg strong)$

For completeness, we define and compute *both* error and *neither* error. However, in practice, since our toy setting contains no spurious correlations other than those due to the weak feature, we find that these two error metrics are at or near zero for all of our experiments (except for a small number of edge cases discussed in §C). Thus, for all results in Section 3, we only show plots of *strong-only* and *weak-only* error, and leave plots of the others to Appendix C.

[2] http://bit.ly/counterexamples

[3] The results shown here used the test error for early-stopping; we have re-run some of our experiments with early-stopping on validation error, and it did not make a difference.

Feature Nickname	Description	Loss AUC	Example
`contains 1`	`1` occurs somewhere in the sequence	0.50	`2 4 11 1 4`
`prefix duplicate`	Sequence begins with a duplicate	0.52	`2 2 11 12 4`
`first-last duplicate`	First number equals last number	0.55	`2 4 11 12 2`
`adjacent duplicate`	Adjacent duplicate is somewhere in the sequence	1.43	`11 12 2 2 4`
`contains first`	First number is elsewhere in the sequence	1.54	`2 11 2 12 4`

Table 1: Features used to instantiate the strong feature in our experimental setup. Features are intended to differ in how hard they are for an LSTM to detect given sequential input. We use the the AUC of the validation loss curve as a heuristic measure of "hardness", as described in Section 2.3.

Interpretation. Intuitively, high *weak-only* error indicates that the model is associating the weak feature with the positive label, whereas high *strong-only* error indicates the model is either failing to detect the strong feature altogether, or is detecting it but failing to associate it with positive label. In practice, we might prioritize these error rates differently in different settings. For example, within work on bias and fairness (Hall Maudslay et al., 2019; Zmigrod et al., 2019), we are primarily targeting *weak-only* error. That is, the primary goal is to ensure that the model does not falsely associate protected attributes with specific labels or outcomes. In contrast, in discussions about improving the robustness of NLP more generally (Elkahky et al., 2018; McCoy et al., 2019), we are presumably targeting *strong-only* error. That is, we are hoping that by lessening the effectiveness of shallow heuristics, we will encourage models to learn deeper, more robust features in their place.

2.5 Data Augmentation

Definition. Adversarial data augmentation aims to reduce the above-described errors by generating new training examples which decouple the strong and weak features. Parallel to the error categories, we can consider two types of counterexamples: ***weak-only* counterexamples** in which the weak feature occurs without the strong feature and the label is 0, and ***strong-only* counterexamples** in which the strong feature occurs without the weak feature and the label is 1.

Interpretation. Again, in terms of practical interpretation, these two types of counterexamples are meaningfully different. In particular, it is likely often the case that *weak-only* counterexamples are easy to obtain, whereas *strong-only* counterexamples are more cumbersome to construct. For example, considering again the case of NLI and the lexical overlap heuristic from McCoy et al. (2019), it is easy to artificially generate *weak-only* counterexamples (p/h pairs with high lexical overlap but which are not in an entailment relation) using a set of well-designed syntactic templates. However, generating good *strong-only* counterexamples (entailed p/h pairs without lexical overlap) is likely to require larger scale human effort (Williams et al., 2018). This difference would likely be exacerbated by realistic problems in which there are many weak features which we may want to remove and/or it is impossible to fully isolate the strong features from all weak features. For example, it may not be possible to decouple the "meaning" of a sentence from all lexical priors. We explore the case of multiple weak features in Section 3.4.

2.6 Limitations

This study is intended to provide an initial framework and intuition for thinking about the relationships between data augmentation and model generalization. We simplify the problem significantly in order to enable controlled and interpretable experiments. As a result, the problems and models studied are several steps removed from the what NLP looks like in practice. In particular, we consider a very small problem (binary classification of sequences of length 5 in which only two variable are not independent) and a very small model (a simple LSTM). Although we provide many additional results in the Appendix to show consistency across different model and task settings, we do not claim that the presented results would hold for more complex models and tasks–e.g. multilayer transformers performing language modeling. Clearly many details of our setup–e.g. which features are easier or harder to detect–would change significantly if we were to change the model architecture and/or training objective to reflect more realistic settings. Exploring whether the overarching trends we observe still hold given such changes would be exciting follow up work.

We also make the assumption that the input to the model contains a single "true" feature which, once extracted, perfectly explains the output. For many of the tasks currently studied in NLP, this assumption arguably never holds, or rather, models only ever get access to correlates of the strong feature. That is, they see text descriptions but never the underlying referents. Thus, for a task like NLI, it might be that the best our models can do is find increasingly strong correlates of "meaning", but may not ever extract "meaning" itself. This is a much larger philosophical debate on which we do not take a stance here. We let it suffice that, in practice, the presented results are applicable to any task in which there exists some stronger (deeper) feature(s) that we prefer the model to use and some weaker (shallower) feature(s) which it may chose to use instead, whether or not those strong features are in fact a "true" feature that determines the label.

3 Results and Discussion

Our primary research questions, reframed in terms of the above terminology, are the following. First, how many counterexamples are needed in order to reduce the model's prediction error? In particular, how is this number influenced by the hardness of the strong features (§3.1), the type of counterexamples added (i.e. *strong-only* vs. *weak-only*) (§3.2), and the size of the training set (§3.3)? Second, given a setting in which multiple weak features exist, does the model prefer to make decisions based on multiple weak features, or rather to extract a single strong feature (§3.4)? We report all results in terms of *strong-only* error and *weak-only* error; results for other error categories are given in Appendix C.

3.1 Effect of Strong Feature's Hardness

We first consider prediction error as a function of the number of counterexamples added and the hardness of detecting the strong feature (with "hardness" defined as in Section 2.3). To do this, we construct an initial training set of 200K examples in which there is perfect co-occurrence between the strong and weak features, with the dataset split evenly between positive examples (e.g. has both strong and weak features) and negative examples (with neither feature). We then vary the number of counterexamples added[4] from 10 ($\ll 0.1\%$ of the training data) to 100K (33% of the training data) and measure the effect on *strong-only* and *weak-only* error. For now, we assume that the counterexamples added are evenly split between *strong-only* and *weak-only* types.

Figure 3 shows the results. We see that the number of counterexamples needed is substantially influenced by the hardness of the strong feature. For example, after only 10 counterexamples are added to training, test error has dropped to near zero when the strong feature is `contains 1` (which is trivial for the model to detect) but remains virtually unchanged for the `contains first` and `adjacent duplicate` features. For the harder features, we don't reach zero error until a third of the training data is composed of counterexamples.

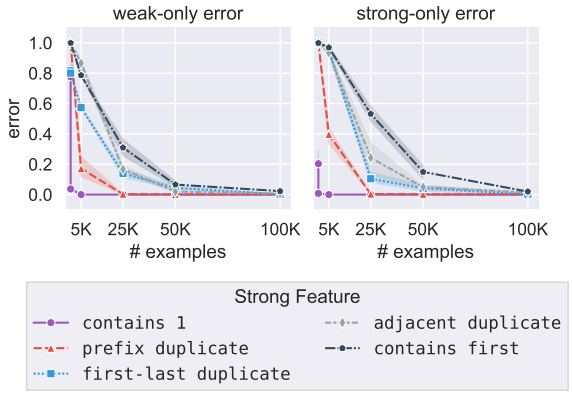

Figure 3: The harder the strong feature, the more counterexamples the model requires to reduce a given error rate. The legend shows models in order from hardest to least hard, where "hardness" is determined by the AUC of the loss curve for a classifier trained to detect the feature (§2.3). We run all experiments over five random seeds. In all plots, the error band is the 95% confidence interval with 1,000 bootstrap iterations.

3.2 Effect of Counterexample Type

We get a better understanding of the model's behavior when looking separately at *strong-only* and *weak-only* errors, considering each as a function of the number and the type (e.g. *strong-only* vs. *weak-only*) of counterexamples added (Figure 4). We see

[4]The results presented assume that the counterexamples are added on top of the training data that is already there, and thus models trained with larger numbers of counterexamples have a slightly larger total training size. We also experimented with adding counterexamples in place of existing training examples so that the total training size remains fixed across all runs. We do not see a meaningful difference in results. Results from the constant-training-size experiments are given in Appendix D.2. We also perform a control experiment to verify that the additional number of training examples itself is not impacting the results in Appendix D.1.

that adding *weak-only* counterexamples leads to improvements in *weak-only* error, but has minimal effect on *strong-only* error. The interesting exception to this is when the strong feature is no harder to represent than the weak feature. In our setting, this happens when the strong feature is `contains 1`, but it is unclear if this pattern would hold for other weak features. In this case, we see that both *strong-only* and *weak-only* error fall to zero after adding only a small number of *weak-only* counterexamples.

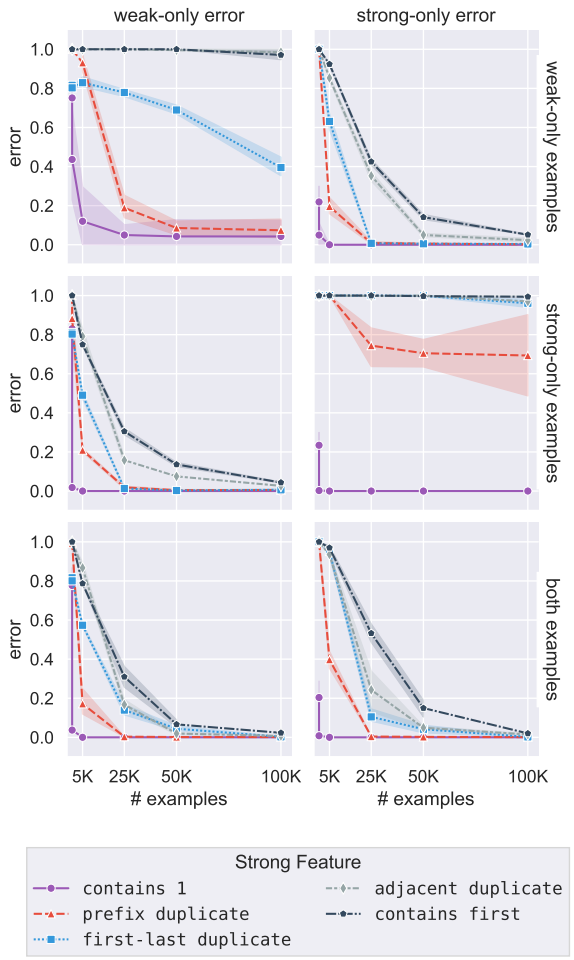

Figure 4: Adding *weak-only* counterexamples improves *weak-only* error but does not affect *strong-only* error. In contrast, adding *strong-only* counterexamples may lead to improvements of both error types (see text for discussion).

In contrast, we see some evidence that adding *strong-only* counterexamples has impact on *weak-only* error as well as on *strong-only* error. The impact on *weak-only* error, however, is limited to the settings in which the strong feature is sufficiently easy to detect. That is, when the strong feature is difficult to detect, adding *strong-only* counterexamples *does* lead the model to detect the strong feature and correctly associate it with the positive label, but *does not* necessarily lead the model to abandon the use of the weak feature in predicting a positive label. This behavior is interesting, since any correct predictions made using the weak feature are by definition redundant with those made using the strong feature, and thus continuing to hold the weak feature can only hurt performance.

3.3 Effect of Training Data Size

In Section 3.1, we observed that, for most of our features, the model did not reach near zero error until 20% or more of its training data was composed of counterexamples. This raises the question: is error rate better modeled in terms of the absolute number of counterexamples added, or rather the fraction of the training data that those counterexamples make up? For example, does adding 5K counterexamples produce the same effect regardless of whether those 5K make up 5% of a 100K training set or 0.05% of a 10M training set? Our intuition motivating this experiment is that larger initial training sets might "dilute" the signal provided by the comparably small set of counterexamples, and thus larger training sets might require substantially more counterexamples to achieve the same level of error.

In Figure 5, we again show both *weak-only* and *strong-only* error as a function of the number of counterexamples added to training, but this time for a range of models trained with different initial training set sizes. Here, for simplicity, we again assume that the counterexamples added are evenly split between the *strong-only* and *weak-only* types. Generally speaking, for most features, we see that our intuition does not hold. That is, increasing the training size while holding the number of counterexamples fixed does not substantially effect either error metric, positively or negatively. That said, there are several noteworthy exceptions to this trend. In particular, we see that when the training set is very large (10M) relative to the number of counterexamples (< 50K), the efficacy of those counterexamples does appear to decrease. We also again see differences depending on the strong feature, with the harder features (`contains first` and `adjacent duplicate`) behaving more in line with the "diluting" intuition described above than the easier features (`first-last duplicate` and `prefix duplicate`).

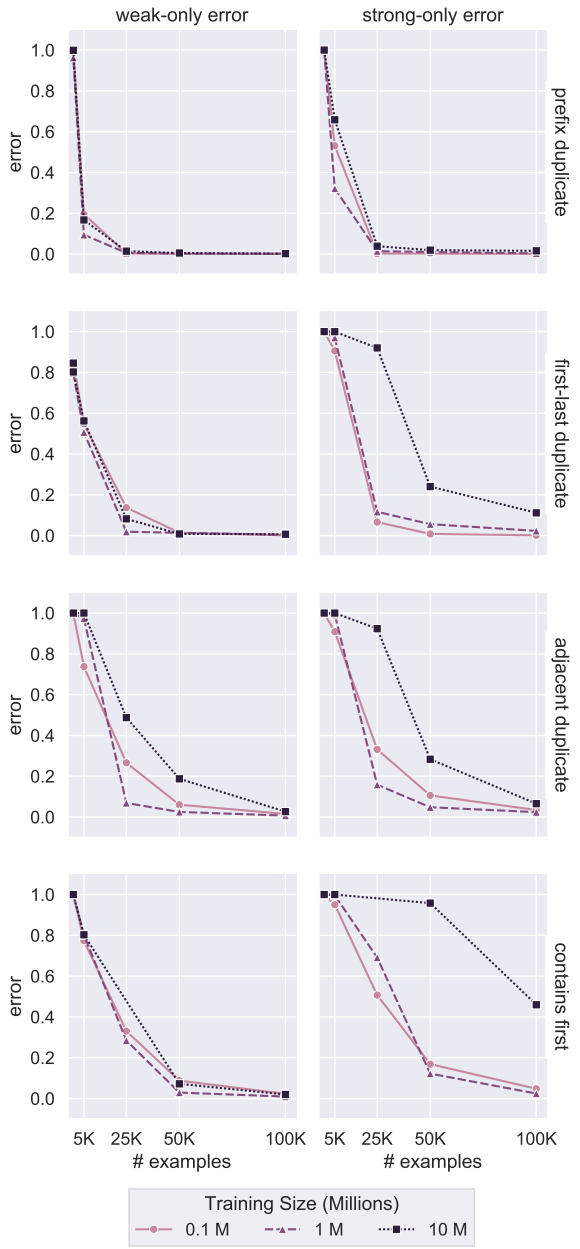

Figure 5: Error rate vs. number of counterexamples added for models trained with different initial training set sizes (100K to 10M). Feature `contains 1` is not shown since it is always learned instantly. In general, increasing the training size while holding the number of counterexamples fixed does not significantly affect the efficacy of those counterexamples, although there are exceptions (see text for discussion).

3.4 Multiple Weak Features

In our experiments so far, we have assumed that there is only one weak feature, which is an unrealistic approximation of natural language data. We therefore relax this assumption and consider the setting in which there are multiple weak features which are correlated with the label. The question is: does the option of minimizing loss by combining multiple weak features lessen the model's willingness to extract strong features?

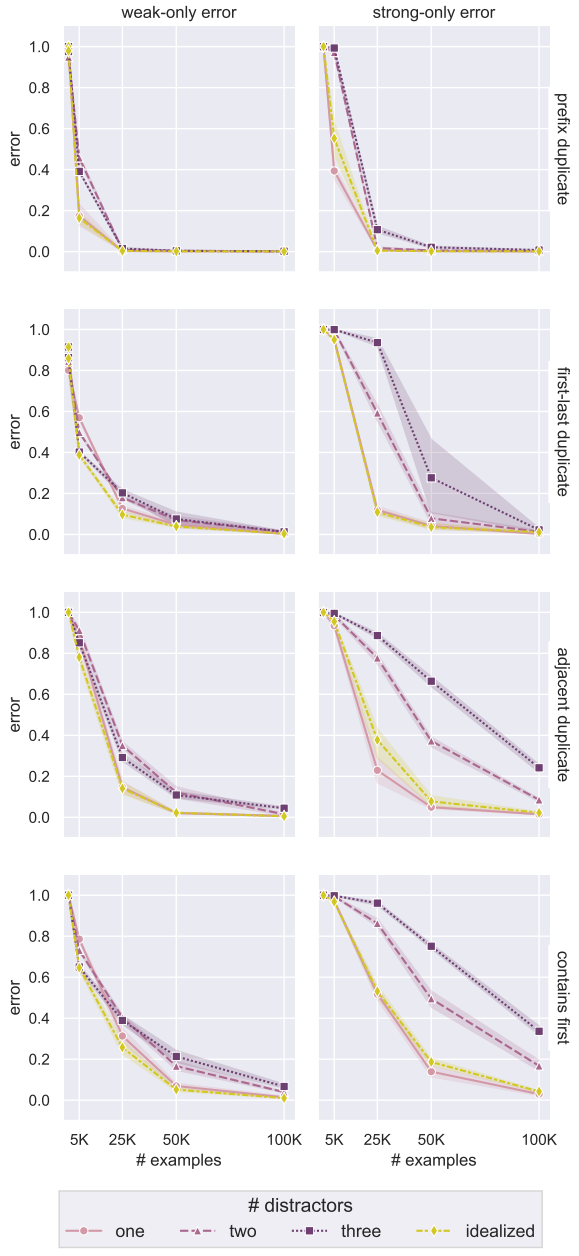

Figure 6: Error rate vs. number of counterexamples added for models trained in settings with different numbers of weak features, assuming that *strong-only* counterexamples can guarantee the removal of at most one weak feature at a time. Initial training size is 200K. Gold line shows performance in an idealized setting in which there are 3 weak feature but *strong-only* counterexamples are "pure", i.e. free from all weak features. When more weak features are present, we see higher *strong-only* error.

Our experimental setup is as follows. We assume k weak features $d_1, \ldots d_k$, each of which

is correlated with the strong feature t, but which are not correlated with each other beyond their correlation with t. As in previous experiments, $P(d_i|t)$ and $P(t|d_i)$ are determined by the number of *strong-only* and *weak-only* counterexamples that have been added, respectively. In the initial training set, before any counterexamples have been added, where $P(t|d_i)$ is 1, and $P(d_i|t)$ is close to $1/2$[5]. We assume that *weak-only* counterexamples can be generated in the same way as before, but that good *strong-only* counterexamples cannot be generated perfectly. For intuition, again consider the NLI setting. It is easy for a person to generate a p/h pair with the label CONTRADICTION, but doing so while meeting the constraints that the sentences exhibit lexical overlap (McCoy et al., 2019) but do not contain any antonym pairs (Dasgupta et al., 2018) or explicit negations (Gururangan et al., 2018) would likely lead to unnatural and unhelpful training sentences. Thus, we assume that we can only reliably remove at most one[6] weak feature at a time. Specifically, we assume that, for a given weak feature d_i, we can generate *strong-only*$_i$ counterexamples which 1) definitely exhibit the strong feature, 2) definitely do not exhibit d_i, and 3) exhibit every other weak features d_j with probability $P(d_j|t)$.

We again plot *strong-only* and *weak-only* error as a function of the number of counterexamples added, assuming we add counterexamples which are evenly split between *weak-only* and *strong-only* types and evenly split among the k weak features. Note that, while our *strong-only* counterexamples are not "pure" (i.e. they might contain weak features other than the one specifically targeted), our *strong-only* error is still computed on examples that are free of *all* weak features. Our *weak-only* error is computed over examples that contain no strong feature and at least one weak features. The results are shown in Figure 6 for $k = \{1, 2, 3\}$.[7] For comparison, we also plot the performance of a model trained in an idealized setting in which "pure" *strong-only* counterexamples are added. We see that, holding the number of counterexamples fixed, the more weak features there are, the higher the *strong-only* error tends to be. We see also that, in an idealized setting in which it is possible to generate "pure" *strong-only* counterexamples, this trend does not hold, and there is no difference between the single weak features and the multiple weak features settings. In terms of *weak-only* error, we see only small increases in error as the number of weak features increases.

Taken together, our interpretation of these results is that access to both *weak-only* counterexamples and to imperfect *strong-only* counterexamples is sufficient for the model to unlearn undesirable weak features–that is, the model does learn that none of the d_i alone causes a positive label, and thus is able to achieve effectively zero *weak-only* error. However, as evidenced by the substantially higher *strong-only* error, the model does not appear to learn to use the strong feature instead. Its worth acknowledging that the model appears to learn some information about the strong feature, as we do see that *strong-only* error falls as we add counterexamples, just more slowly for larger values of k. This trend might be taken as evidence that the high *strong-only* error is not due to a failure to detect the feature (as was the case in Figure 4 when only *weak-only* errors were added), but rather a failure to consistently associate the strong feature with the positive label.

4 Related Work

4.1 Adversarial Data Augmentation

A wave of recent work has adopted a strategy of constructing evaluation sets composed of "adversarial examples" (a.k.a. "challenge examples" or "probing sets") in order to analyze and expose weaknesses in the decision procedures learned by neural NLP models (Jia and Liang, 2017; Glockner et al., 2018; Dasgupta et al., 2018; Gururangan et al., 2018; Poliak et al., 2018b, and others). Our work is motivated by the subsequent research that has begun to ask whether adding such challenge examples to a model's training data could help to improve robustness. This work has been referred to by a number of names including "adversarial", "counterfactual", and "targeted" data augmentation. In particular, Liu et al. (2019) show that fine-tuning on small challenge sets can sometimes (though not always) help models perform better. Similar approaches have been explored for handling noun-verb ambiguity in syntactic parsing (Elkahky et al.,

[5] We independently sample each feature with probability $1/2$ but then resample if the example does not contain any weak features. This is done such that the reported number of counterexamples is correct.

[6] We ran experiments in which it was possible to remove more than one weak feature at a time (but still fewer than k) and did not see interestingly different trends from the at-most-one setting.

[7] Given our sequence length of 5, we were unable to generate results for larger values of k without interfering with our ability to include the strong feature.

2018), improving NLI models' handling of syntactic (McCoy et al., 2019) and semantic (Poliak et al., 2018a) phenomena, and mitigating gender biases in a range of applications (Zmigrod et al., 2019; Zhao et al., 2018, 2019; Hall Maudslay et al., 2019; Lu et al., 2018).

4.2 Adversarial Robustness

Adversarial robustness concerns whether a model produces the same output when the input has been perturbed such that its underlying semantics are unchanged. In computer vision, these perturbations might be low-level noise added to the pixels. But defining the set of valid perturbations is open problem in NLP, where small changes in surface-form could dramatically change the underlying meaning of an utterance (removing 'not', for example). There has, however, been work on constructing perturbations by replacing words in an utterance with their synonyms (Alzantot et al., 2018; Hsieh et al., 2019; Jia et al., 2019) and generating new sentences via paraphrases (Ribeiro et al., 2018; Iyyer et al., 2018). In particular, Jia et al. (2019) derive bounds on error within this well-defined set of perturbations. In the context of *evaluation* of generated perturbations, recent works have discussed the extent to which they induce models to make a wrong prediction (Ribeiro et al., 2018; Iyyer et al., 2018; Hsieh et al., 2019; Jia et al., 2019) or change their output (Alzantot et al., 2018). Hsieh et al. (2019) also analyze these perturbations' effect on attention weights.

Outside of NLP, Ilyas et al. (2019) make a distinction between useful features (that generalize well) and those that are robustly-useful (that generalize well, even if an example is adversarially perturbed). They are able to create a data set with only robust features. And related to this work by Ilyas et al. (2019), there's been significant recent interest in training models such that they're robust to adversarial examples and in building adversarial datasets that foil such defenses. Highlighting just two recent papers, Madry et al. (2017) describe training that's robust against adversaries with access to a model's gradients, while Athalye et al. (2018) show that many defenses are "obfuscating" their gradients in a way that can be exploited.

4.3 Encoding Structure in NLP Models

Another related body of work focuses on understanding what types of features are extracted by neural language models, in particular looking for evidence that SOTA models go beyond bag-of-words representations and extract "deeper" features about linguistic structure. Work in this vein has produced evidence that pretrained language models encode knowledge of syntax, using a range of techniques including supervised "diagnostic classifiers" (Tenney et al., 2019; Conneau et al., 2018; Hewitt and Manning, 2019), classification performance on targeted stimuli (Linzen et al., 2016; Goldberg, 2019), attention maps/visualizations (Voita et al., 2019; Serrano and Smith, 2019), and relational similarity analyses (Chrupała and Alishahi, 2019). Our work contributes to this literature by focusing on a toy problem and asking under what conditions we might expect deeper features to be extracted, focusing in particular on the role that the training distribution plays in encouraging models to learn deeper structure. Related in spirit to our toy data approach is recent work which attempts to quantify how much data a model should need to learn a given deeper feature (Geiger et al., 2019). Still other related work explores ways for encouraging models to learn structure which do not rely on data augmentation, e.g. by encoding inductive biases into model architectures (Bowman et al., 2015; Andreas et al., 2016) in order to make "deep" features more readily extractable, or by designing training objectives that incentivize the extraction of specific features (Swayamdipta et al., 2017; Niehues and Cho, 2017). Exploring the effects these modeling changes on the results presented in this paper is an exciting future direction.

4.4 Generalization of Neural Networks

Finally, this work relates to a still larger body of work in which aims to understand feature representation and generalization in neural networks in general. Mangalam and Prabhu (2019) show that neural networks learn "easy" examples (as defined by their learnability by shallow ML models) before they learn "hard" examples. Zhang et al. (2016) and Arpit et al. (2017) show that neural networks with good generalization performance can nonetheless easily memorize noise of the same size, suggesting that, when structure does exist in the data, models might have some inherent preference to learn general features even though memorization is an equally available option. Zhang et al. (2019) train a variety of over-parameterized models on the identity mapping and show that some fail entirely while others learn a generalizable identify func-

tion, suggesting that different architectures have different tendencies for learning structure vs. memorizing. Finally, there is ongoing theoretical work which attempts to characterize the ability of overparameterized networks to generalize in terms of complexity (Neyshabur et al., 2019) and implicit regularization (Blanc et al., 2019).

5 Conclusion

We propose a framework for simulating the effects of data augmentation in NLP and use it to explore how training on counterexamples impacts model generalization. Our results suggest that adding counterexamples in order to encourage a model to "unlearn" weak features is likely to have the immediately desired effect (the model will perform better on examples that look similar to the generated counterexamples), but the model is unlikely to shift toward relying on stronger features in general. Specifically, in our experiments, the models trained on counterexamples still fail to correctly classify examples which contain only the strong feature. We see also that data augmentation may become less effective as the underlying strong features become more difficult to extract and as the number of weak features in the data increases.

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 233–242. JMLR.org.

Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. 2019. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. *arXiv preprint arXiv:1904.09080*.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21, Beijing, China. Association for Computational Linguistics.

Grzegorz Chrupała and Afra Alishahi. 2019. Correlating neural and symbolic representations of language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2952–2962, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. 2018. Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

Ali Elkahky, Kellie Webster, Daniel Andor, and Emily Pitler. 2018. A challenge set and methods for noun-verb ambiguity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2562–2572, Brussels, Belgium. Association for Computational Linguistics.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2019. Posing fair generalization tasks for natural language inference. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4484–4494, Hong Kong, China. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

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

Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell, and Simone Teufel. 2019. It's all in the name: Mitigating gender bias with name-based counterfactual data substitution. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5266–5274, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics.

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

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2171–2179, Minneapolis, Minnesota. Association for Computational Linguistics.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2018. Gender bias in neural natural language processing. *arXiv preprint arXiv:1807.11714*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Karttikeya Mangalam and Vinay Uday Prabhu. 2019. Do deep neural networks learn shallow learnable examples first?

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

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. 2019. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*, pages 80–89, Copenhagen, Denmark. Association for Computational Linguistics.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018a. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018b. Hypothesis only baselines in natural language inference. *arXiv preprint arXiv:1805.01042*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, and Yoram Singer. 2019. Identity crisis: Memorization and generalization under extreme overparameterization.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. *arXiv preprint arXiv:1904.03310*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Ran Zmigrod, Sebastian J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

A Effect of Hyperparameter Settings

We vary several model and task settings, and we find that the models exhibit similar behavior as in the default case. The default settings are summarized in Table 2. We run all experiments over five random seeds, arbitrarily set to $42, 43, 44, 45, 46$. In all plots the error band is the 95% confidence interval with 1,000 bootstrap iterations. We do not average over all the random seeds for experiments with dataset size because of the higher computational cost.

We find that batch size (16, 32, 64) and dropout (0.0, 0.01, 0.1, 0.5) within the MLP do not affect model performance. We also find that embedding and hidden sizes (50, 125, 250, 500) do not significantly impact results, except that performance is worse for 500. This is likely because more training data is needed for these higher capacity models. We don't include figures for these experiments for the sake of conciseness. We present results below in Section A.1 for different vocabulary sizes, and we find that the results don't change significantly, though the model is unable to learn the strong feature for the smallest vocabulary size.

Hyperparameter	**Value**
Model Settings	
optimizer	Adam
early stopping patience	5
batch size	64
number of LSTM layers	1
hidden dimensionality	250
embedding dimensionality	250
initial learning rate	0.001
dropout	0
Task Settings	
vocabulary size	50K
training size	10K
sequence length	5

Table 2: Hyperparameter settings. We set hyperparamters as default values for our experiments above. Separately, we investigate various values for batch size, dropout, and embedding and vocab sizes and we find that they do not significantly affect the results, with the exception of models with high embedding and hidden sizes. We find the same for vocabulary size (discussed below in A.1).

A.1 Vocabulary Size

We re-run the main experiment – in which we vary the number of adversarial counterexamples – at different vocabulary sizes. The results are shown in Figure 7. We observe similar results when the vocabulary size is 5K as for the default of 50K. The models learn the strong features to some extent as we increase the number of counterexamples, and more counterexamples are needed for the model to generalize well when the strong feature is harder. However, we note that `first-last duplicate` is harder than the other strong geatures at this vocabulary size. We aren't sure why this is the case because we see relative hardness similar to the default case in classification experiments with vocabulary size of 5K (the model is trained on identifying whether the strong feature holds as in Figure 2). We also note that in the same classification experiments with a vocabulary size of 0.5K, the model was not able to learn any of these strong features, which explains why the model didn't learn the strong features at any number of counterexamples.

B Complicating the Strong Feature

In a real-world setting, there often isn't an underlying strong feature that exactly predicts the label. We consider two variants of the experimental setup that weaken this assumption. Specifically, we introduce label noise, and the case when the strong feature is a union of multiple other features.

B.1 Random Label Noise

For some noise level ϵ, we independently flip each label (0 becomes 1, and 1 becomes 0) with probability ϵ. The results are in Figure 8. The trends are fairly resistant to noise. The model continues to switch to learning the strong feature (though it takes slightly more adversarial counterexamples with more noise), and we continue to observe the relationship between the features' hardness and prediction error as a function of the number of adversarial counterexamples.

B.2 Multiple Strong Features

We relax the assumption that there's a single feature that exactly predicts the label, and we instead consider k strong features $t_1, \ldots t_k$ that occur one-at-a-time with equal probability in examples for which the strong feature holds (i.e. examples for which both the strong and weak features hold and *strong-*

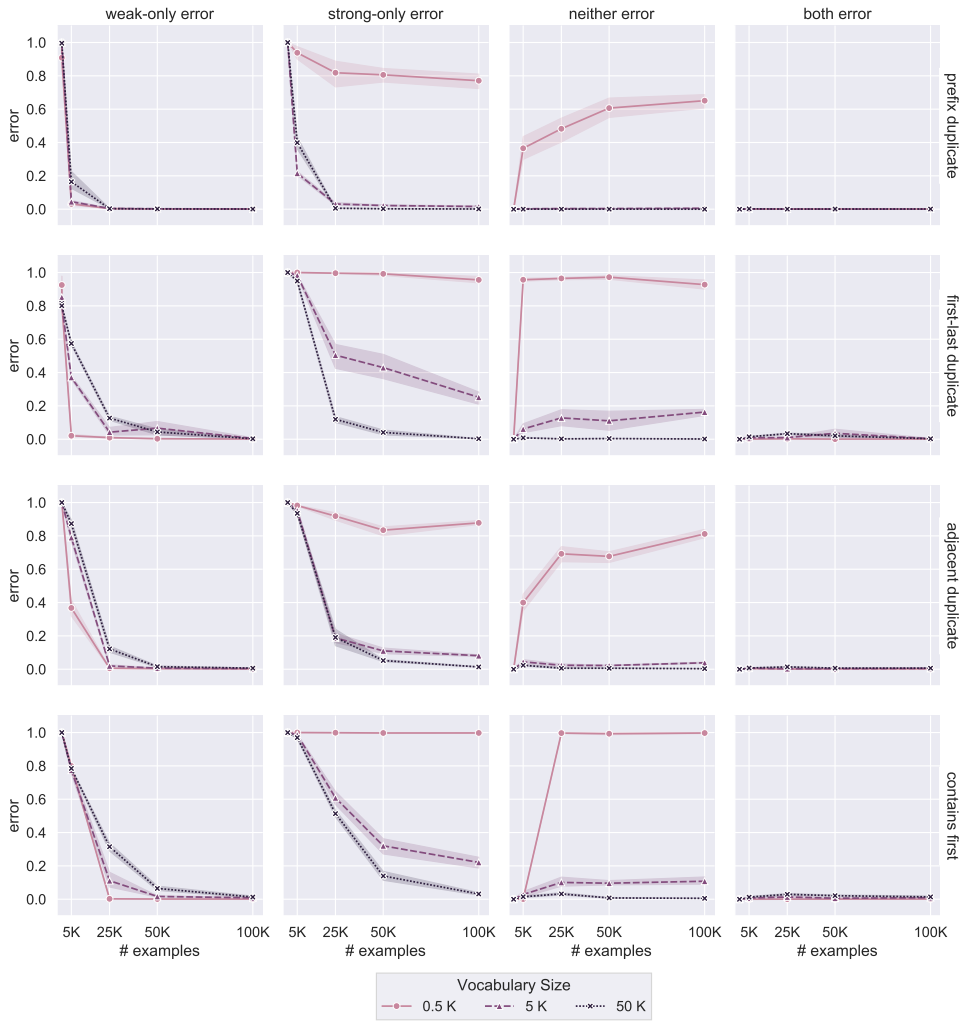

Figure 7: Vocabulary size. The model isn't able to learn the strong features at a vocabulary size of 0.5K, but we observe similar behavior at 5K and 50K.

only adversarial counterexamples). Intuitively, the strong feature becomes $t_1 \vee t_2 \vee \ldots t_n$. Figure 9 shows the results. We find that the model, in most cases, switches from relying on the weak feature to learning to use the collection of strong features. We observe that collections of harder features take more adversarial counterexamples to achieve low error than collections of easy features, which is consistent with our previous results. Interestingly, in two of the three cases, we find that a collection of features might take more adversarial counterexamples to achieve low error than any one of the features in the collection. However, in the third case (the rightmost plot in Figure 9), the error of `contains first` exceeds that of the combined strong feature.

C Other Error Metrics

We include additional figures for our main results – hardness (Figure 10), *strong-only* and *weak-only* counterexamples (Figure 11), training size (Figure 12), multiple weak feature (Figure 13) – that present the error for the other two regions of the test data (where both or neither of the features hold). With a single exception, we observe that adding counterexamples doesn't lead the model to generalize worse on data that isn't adversarial, which means that data augmentation helps with overall test error, in addition to test error on adversarial examples. This is expected; if the model switches from using the weak feature to using the strong feature, its performance shouldn't change on examples where both or neither feature holds. However, we note that error on these examples increases (and then decreases) for `first-last duplicate`

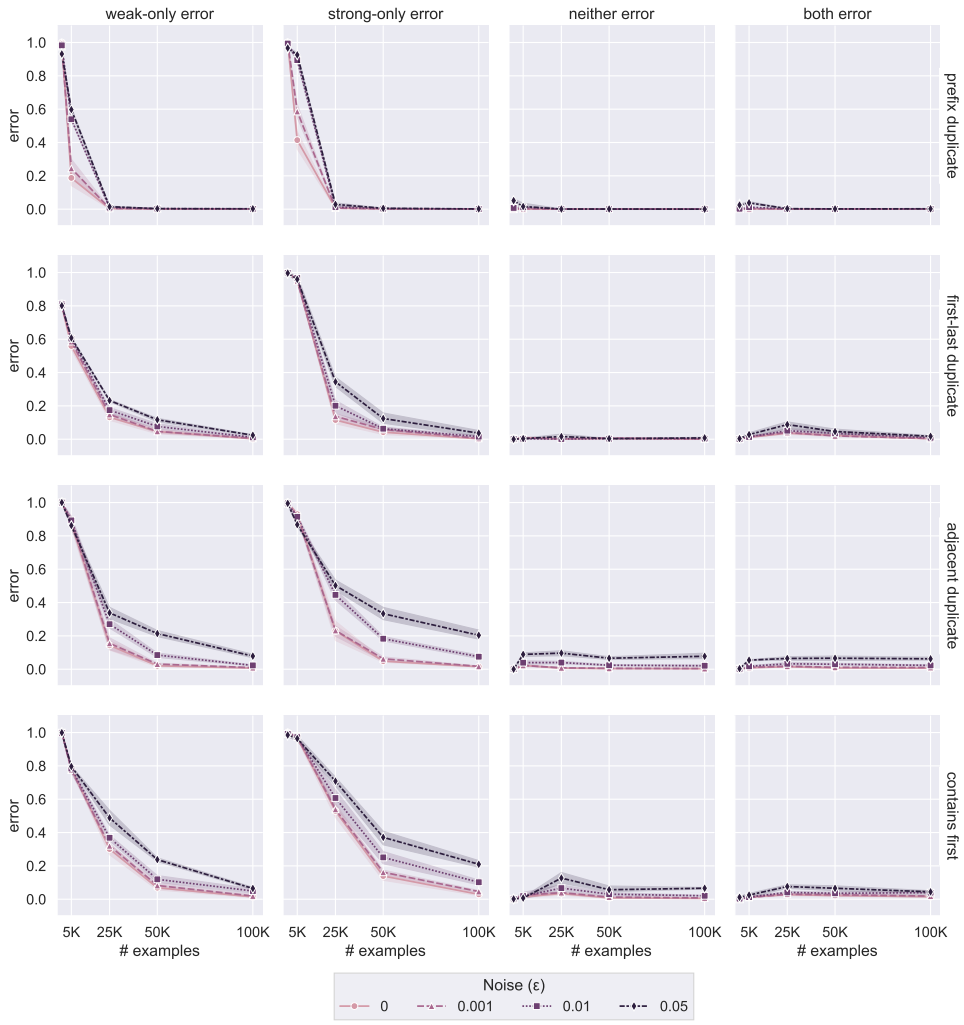

Figure 8: Noise. We observe that adding noise does not affect the results, though more noise seems to make the model less prone to learning the strong feature.

and `contains first` at a training size of 10M (Figure 12).

D Controls for Training Size

D.1 Adding Non-adversarial Examples

When adding adversarial counterexamples, the total number of training examples also increases. We control for this change by showing here that additional "default" (or non-adversarial) training examples do not help the model on either *weak-only* or *strong-only* error. Figure 14 shows that the addition of these examples does not impact the results. Therefore, it matters that added examples are counterexamples; the model doesn't improve simply because there's more data.

D.2 Fixed Training Size

We've shown above that more training data without counterexamples is not sufficient to induce the model to use the strong feature in classification; see Figure 15. However, one might argue that in the presence of some counterexamples, more training data is helpful, whether or not it's adversarial. This would make it hard to disentangle the role of more training data with that of an increased number of counterexamples. Here, we fix training size as we add counterexamples (meaning there are fewer non-adversarial examples) and we observe similar results as in our main experiments above (Figure 3). Naturally, this is not the case for extreme numbers of counterexamples: if we remove all non-adversarial examples, the model is negatively impacted. But these results – taken together with those above – indicate that the benefits of adding counterexamples in general (§3.1) and increasing the number of counterexamples (these results) should not be attributed to the larger training size.

Figure 9: Multiple strong features. We find similar trends as for a single strong feature. Collections of harder features require more counterexamples to achieve low generalization error than collections of easier features. We also find that collections of features might take more counterexamples to achieve low generalization than any individual feature in the collection.

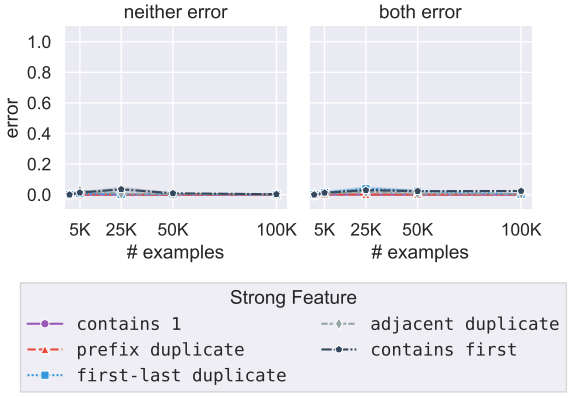

Figure 10: Hardness. Other error cases (*neither* left and *both* right) for Figure 3.

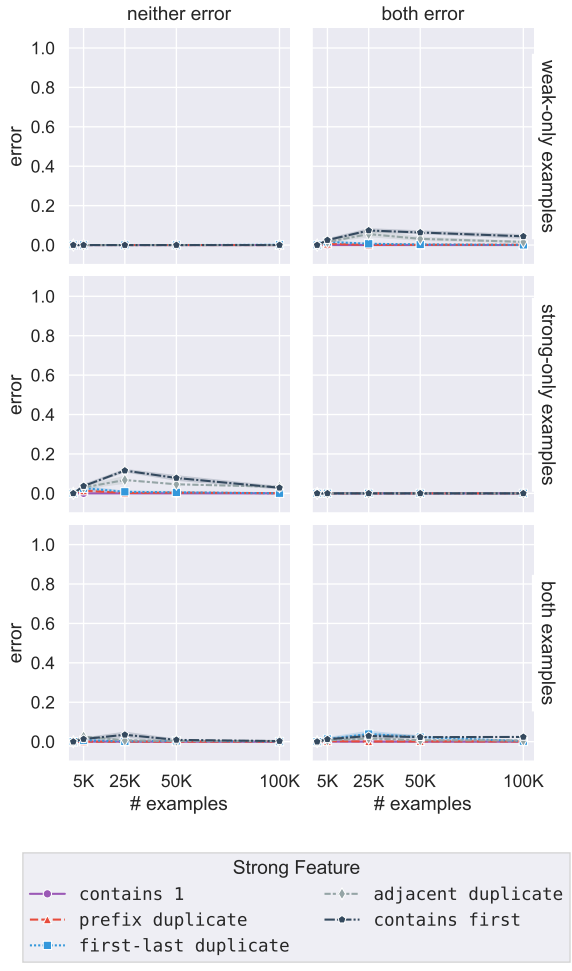

Figure 11: Counterexamples of varying types. Other error cases (*neither* left and *both* right) for Figure 4.

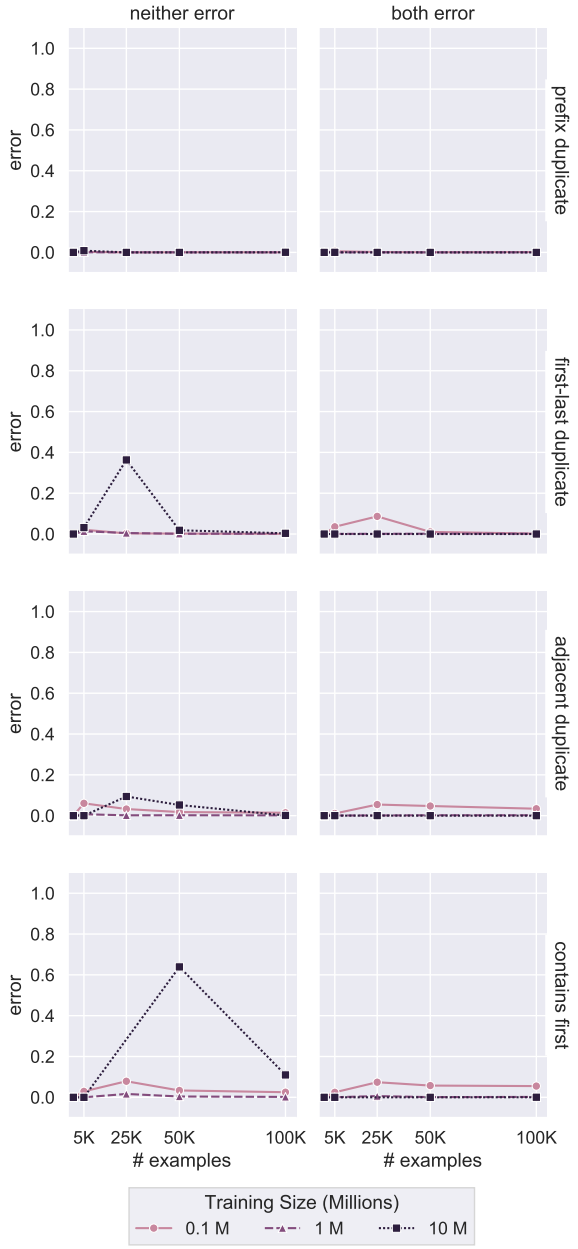

Figure 12: Changing training size. Other error cases (*neither* left and *both* right) for Figure 5.

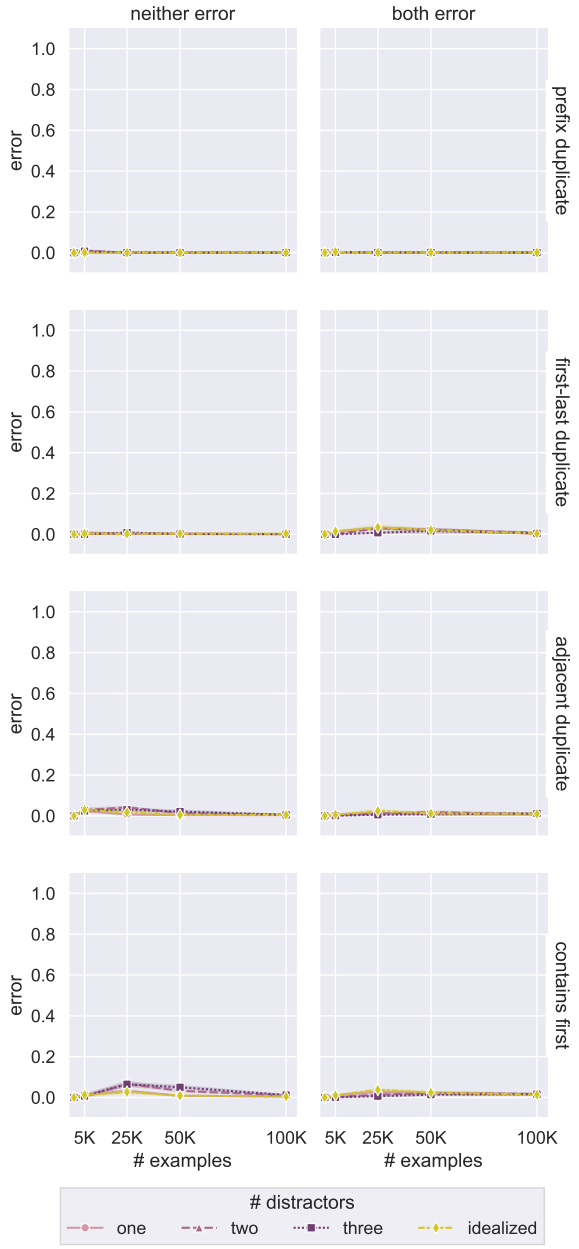

Figure 13: Multiple weak features. Other error cases (*neither* left and *both* right) for Figure 6.

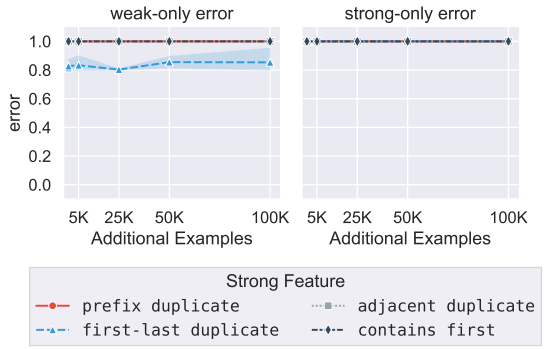

Figure 14: When adding adversarial counterexamples, the total number of training examples also increases. We control for this change by showing here that additional "default" training examples do not help the model on either *weak-only* or *strong-only* error. This is unsurprising given our previously shown results.

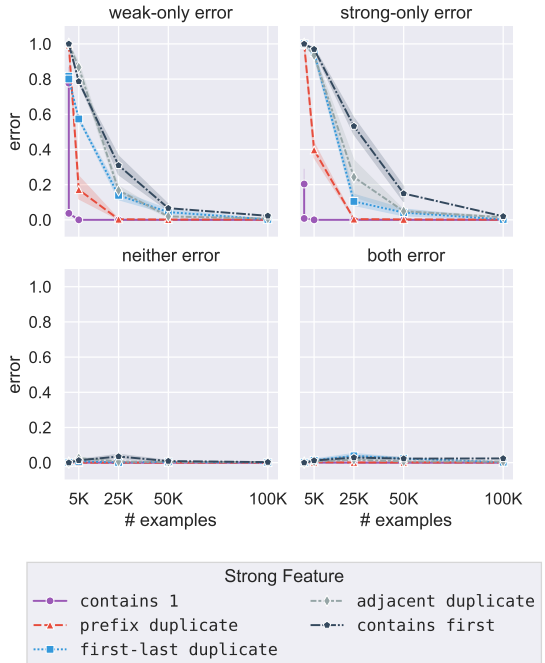

Figure 15: Static training size; counterexamples replace training examples.