AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples

Dongyeop Kang[1] **Tushar Khot**[2] **Ashish Sabharwal**[2] **Eduard Hovy**[1]
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Allen Institute for Artificial Intelligence, Seattle, WA, USA
{dongyeok,hovy}@cs.cmu.edu {tushark,ashishs}@allenai.org

Abstract

We consider the problem of learning textual entailment models with limited supervision (5K-10K training examples), and present two complementary approaches for it. First, we propose knowledge-guided adversarial example generators for incorporating large lexical resources in entailment models via only a handful of rule templates. Second, to make the entailment model—a discriminator—more robust, we propose the first GAN-style approach for training it using a natural language example generator that iteratively adjusts based on the discriminator's performance. We demonstrate effectiveness using two entailment datasets, where the proposed methods increase accuracy by 4.7% on SciTail and by 2.8% on a 1% training sub-sample of SNLI. Notably, even a single hand-written rule, NEGATE, improves the accuracy on the negation examples in SNLI by 6.1%.

1 Introduction

The impressive success of machine learning models on large natural language datasets often does not carry over to moderate training data regimes, where models often struggle with infrequently observed patterns and simple adversarial variations. A prominent example of this phenomenon is *textual entailment*, the fundamental task of deciding whether a *premise* text entails ($\models$) a *hypothesis* text. On certain datasets, recent deep learning entailment systems (Parikh et al., 2016; Wang et al., 2017; Gong et al., 2018) have achieved close to human level performance. Nevertheless, the problem is far from solved, as evidenced by how easy it is to generate minor adversarial examples that break even the best systems. As Table 1 illustrates, a state-of-the-art neural system for this task, namely the Decomposable Attention Model (Parikh et al., 2016), fails when faced with simple linguistic phenomena such as negation, or a re-ordering of words. This is not unique to a particular model or task. Minor adversarial examples have also been found to easily break neural systems on other linguistic tasks such as reading comprehension (Jia and Liang, 2017).

Table 1: Failure examples from the SNLI dataset: negation (Top) and re-ordering (Bottom). **P** is premise, **H** is hypothesis, and **S** is prediction made by an entailment system (Parikh et al., 2016).

P: The dog did not eat all of the chickens.
H: The dog ate all of the chickens.
S: entails (score 56.5%)
P: The red box is in the blue box.
H: The blue box is in the red box.
S: entails (score 92.1%)

A key contributor to this brittleness is the use of specific datasets such as SNLI (Bowman et al., 2015) and SQuAD (Rajpurkar et al., 2016) to drive model development. While large and challenging, *these datasets also tend to be homogeneous*. E.g., SNLI was created by asking crowd-source workers to generate entailing sentences, which then tend to have limited linguistic variations and annotation artifacts (Gururangan et al., 2018). Consequently, models overfit to sufficiently repetitive patterns—and sometimes idiosyncrasies—in the datasets they are trained on. They fail to cover long-tail and rare patterns in the training distribution, or linguistic phenomena such as negation that would be obvious to a layperson.

To address this challenge, we propose to *train textual entailment models more robustly using ad-*

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 2418–2428
Melbourne, Australia, July 15 - 20, 2018.

versarial examples generated in two ways: (a) by incorporating knowledge from large linguistic resources, and (b) using a sequence-to-sequence neural model in a GAN-style framework.

The motivation stems from the following observation. While deep-learning based textual entailment models lead the pack, they generally do not incorporate intuitive rules such as negation, and ignore large-scale linguistic resources such as PPDB (Ganitkevitch et al., 2013) and WordNet (Miller, 1995). These resources could help them generalize beyond specific words observed during training. For instance, while the SNLI dataset contains the pattern *two men* $\models$ *people*, it does not contain the analogous pattern *two dogs* $\models$ *animals* found easily in WordNet.

Effectively integrating simple rules or linguistic resources in a deep learning model, however, is challenging. Doing so directly by substantially adapting the model architecture (Sha et al., 2016; Chen et al., 2018) can be cumbersome and limiting. Incorporating such knowledge indirectly via modified word embeddings (Faruqui et al., 2015; Mrkšić et al., 2016), as we show, can have little positive impact and can even be detrimental.

Our proposed method, which is task-specific but model-independent, is inspired by data-augmentation techniques. We generate new training examples by applying knowledge-guided rules, via only a handful of rule templates, to the original training examples. Simultaneously, we also use a sequence-to-sequence or seq2seq model for each entailment class to generate new hypotheses from a given premise, adaptively creating new adversarial examples. These can be used with any entailment model without constraining model architecture.

We also introduce the first approach to train a robust entailment model using a Generative Adversarial Network or GAN (Goodfellow et al., 2014) style framework. We iteratively improve both the entailment system (the *discriminator*) and the differentiable part of the data-augmenter (specifically the neural *generator*), by training the generator based on the discriminator's performance on the generated examples. Importantly, unlike the typical use of GANs to create a strong generator, we use it as a mechanism to create a strong and robust discriminator.

Our new entailment system, called AdvEntuRe, demonstrates that in the moderate data regime, adversarial iterative data-augmentation via only a handful of linguistic rule templates can be surprisingly powerful. Specifically, we observe 4.7% accuracy improvement on the challenging SciTail dataset (Khot et al., 2018) and a 2.8% improvement on 10K-50K training subsets of SNLI. An evaluation of our algorithm on the negation examples in the test set of SNLI reveals a 6.1% improvement from just a single rule.

2 Related Work

Adversarial example generation has recently received much attention in NLP. For example, Jia and Liang (2017) generate adversarial examples using manually defined templates for the SQuAD reading comprehension task. Glockner et al. (2018) create an adversarial dataset from SNLI by using WordNet knowledge. Automatic methods (Iyyer et al., 2018) have also been proposed to generate adversarial examples through paraphrasing. These works reveal how neural network systems trained on a large corpus can easily break when faced with carefully designed unseen adversarial patterns at test time. Our motivation is different. We use adversarial examples at training time, in a data augmentation setting, to train a more robust entailment discriminator. The generator uses explicit knowledge or hand written rules, and is trained in a end-to-end fashion along with the discriminator.

Incorporating external rules or linguistic resources in a deep learning model generally requires substantially adapting the model architecture (Sha et al., 2016; Liang et al., 2017; Kang et al., 2017). This is a model-dependent approach, which can be cumbersome and constraining. Similarly non-neural textual entailment models have been developed that incorporate knowledge bases. However, these also require model-specific engineering (Raina et al., 2005; Haghighi et al., 2005; Silva et al., 2018).

An alternative is the model- and task-independent route of incorporating linguistic resources via word embeddings that are *retro-fitted* (Faruqui et al., 2015) or *counter-fitted* (Mrkšić et al., 2016) to such resources. We demonstrate, however, that this has little positive impact in our setting and can even be detrimental. Further, it is unclear how to incorporate knowledge sources into advanced representations such as contextual embeddings (McCann et al.,

2017; Peters et al., 2018). We thus focus on a task-specific but model-independent approach.

Logical rules have also been defined to label existing examples based on external resources (Hu et al., 2016). Our focus here is on generating *new* training examples.

Our use of the GAN framework to create a better discriminator is related to CatGANs (Wang and Zhang, 2017) and TripleGANs (Chongxuan et al., 2017) where the discriminator is trained to classify the original training image classes as well as a new 'fake' image class. We, on the other hand, generate examples belonging to the same classes as the training examples. Further, unlike the earlier focus on the vision domain, this is the first approach to train a discriminator using GANs for a natural language task with discrete outputs.

3 Adversarial Example Generation

We present three different techniques to create adversarial examples for textual entailment. Specifically, we show how external knowledge resources, hand-authored rules, and neural language generation models can be used to generate such examples. Before describing these generators in detail, we introduce the notation used henceforth.

We use lower-case letters for single instances (e.g., x, p, h), upper-case letters for sets of instances (e.g., X, P, H), blackboard bold for models (e.g., $\mathbb{D}$), and calligraphic symbols for discrete spaces of possible values (e.g., class labels $\mathcal{C}$). For the textual entailment task, we assume each example is represented as a triple (p, h, c), where p is a premise (a natural language sentence), h is a hypothesis, and c is an entailment label: (a) *entails* ($\sqsubseteq$) if h is true whenever p is true; (b) *contradicts* ($\curlywedge$) if h is false whenever p is true; or (c) *neutral* (#) if the truth value of h cannot be concluded from p being true.[1]

We will introduce various example generators in the rest of this section. Each such generator, $\mathbb{G}_\rho$, is defined by a partial function f_ρ and a label g_ρ. If a sentence s has a certain property required by f_ρ (e.g., contains a particular string), f_ρ transforms it into another sentence s' and g_ρ provides an entailment label from s to s'. Applied to a sentence s, $\mathbb{G}_\rho$ thus either "fails" (if the pre-requisite isn't met) or generates a new entailment example triple, $\big(s, f_\rho(s), g_\rho\big)$. For instance, consider the generator

[1]The symbols are based on Natural Logic (Lakoff, 1970) and use the notation of MacCartney and Manning (2012).

Source	ρ	f_ρ(s)	g_ρ
Knowledge Base, $\mathbb{G}^{\text{KB}}$			
WordNet	hyper(x, y)	Replace x with y in s	$\sqsubseteq$
	anto(x, y)		$\curlywedge$
	syno(x, y)		$\sqsubseteq$
PPDB	$x \equiv y$		$\sqsubseteq$
SICK	$c(x, y)$		c
Hand-authored, $\mathbb{G}^{\text{H}}$			
Domain knowledge	NEG	NEGATE(s)	$\curlywedge$
Neural Model, $\mathbb{G}^{\text{s2s}}$			
Training data	(s2s, c)	$\mathbb{G}^{\text{s2s}}_c(s)$	c

Table 2: Various generators $\mathbb{G}_\rho$ characterized by their source, (partial) transformation function f_ρ as applied to a sentence s, and entailment label g_ρ

for ρ:=hypernym(car, vehicle) with the (partial) transformation function f_ρ:="Replace *car* with *vehicle*" and the label g_ρ:=*entails*. f_ρ would fail when applied to a sentence not containing the word "car". Applying f_ρ to the sentence s="*A man is driving the car*" would generate s'="*A man is driving the vehicle*", creating the example $(s, s', \textit{entails})$.

The seven generators we use for experimentation are summarized in Table 2 and discussed in more detail subsequently. While these particular generators are simplistic and one can easily imagine more advanced ones, we show that training using adversarial examples created using even these simple generators leads to substantial accuracy improvement on two datasets.

3.1 Knowledge-Guided Generators

Large knowledge-bases such as WordNet and PPDB contain lexical equivalences and other relationships highly relevant for entailment models. However, even large datasets such as SNLI generally do not contain most of these relationships in the training data. E.g., that *two dogs* entails *animals* isn't captured in the SNLI data. We define simple generators based on lexical resources to create adversarial examples that capture the underlying knowledge. This allows models trained on these examples to learn these relationships.

As discussed earlier, there are different ways of incorporating such symbolic knowledge into neural models. Unlike task-agnostic ways of approaching this goal from a word embedding perspective (Faruqui et al., 2015; Mrkšić et al., 2016)

or the model-specific approach (Sha et al., 2016; Chen et al., 2018), we use this knowledge to generate task-specific examples. This allows any entailment model to learn how to use these relationships *in the context of the entailment task*, helping them outperform the above task-agnostic alternative.

Our knowledge-guided example generators, $\mathbb{G}_\rho^{\text{KB}}$, use lexical relations available in a knowledge-base: $\rho := r(x, y)$ where the relation r (such as synonym, hypernym, etc.) may differ across knowledge bases. We use a simple (partial) transformation function, $f_\rho(s)$:="Replace x in s with y", as described in an earlier example. In some cases, when part-of-speech (POS) tags are available, the partial function requires the tags for x in s and in $r(x, y)$ to match. The entailment label g_ρ for the resulting examples is also defined based on the relation r, as summarized in Table 2.

This idea is similar to Natural Logic Inference or NLI (Lakoff, 1970; Sommers, 1982; Angeli and Manning, 2014) where words in a sentence can be replaced by their hypernym/hyponym to produce entailing/neutral sentences, depending on their context. We propose a context-agnostic use of lexical resources that, despite its simplicity, already results in significant gains. We use three sources for generators:

WordNet (Miller, 1995) is a large, hand-curated, semantic lexicon with synonymous words grouped into *synsets*. Synsets are connected by many semantic relations, from which we use *hyponym* and *synonym* relations to generate entailing sentences, and *antonym* relations to generate contradicting sentences[2]. Given a relation $r(x, y)$, the (partial) transformation function f_ρ is the POS-tag matched replacement of x in s with y, and requires the POS tag to be noun or verb. NLI provides a more robust way of using these relations based on context, which we leave for future work.

PPDB (Ganitkevitch et al., 2013) is a large resource of lexical, phrasal, and syntactic paraphrases. We use 24,273 lexical paraphrases in their smallest set, PPDB-S (Pavlick et al., 2015), as equivalence relations, $x \equiv y$. The (partial) transformation function f_ρ for this generator is POS-tagged matched replacement of x in s with y, and the label g_ρ is *entails*.

SICK (Marelli et al., 2014) is dataset with entailment examples of the form (p, h, c), created to evaluate an entailment model's ability to capture compositional knowledge via hand-authored rules. We use the 12,508 patterns of the form $c(x, y)$ extracted by Beltagy et al. (2016) by comparing sentences in this dataset, with the property that for each SICK example (p, h, c), replacing (when applicable) x with y in p produces h. For simplicity, we ignore positional information in these patterns. The (partial) transformation function f_ρ is replacement of x in s with y, and the label g_ρ is c.

3.2 Hand-Defined Generators

Even very large entailment datasets have no or very few examples of certain otherwise common linguistic constructs such as negation,[3] causing models trained on them to struggle with these constructs. A simple model-agnostic way to alleviate this issue is via a negation example generator whose transformation function $f_\rho(s)$ is NEGATE(s), described below, and the label g_ρ is *contradicts*.

NEGATE(S): If s contains a 'be' verb (e.g., is, was), add a "not" after the verb. If not, also add a "did" or "do" in front based on its tense. E.g., change "A person is crossing" to "A person is not crossing" and "A person crossed" to "A person did not cross." While many other rules could be added, we found that this single rule covered a majority of the cases. Verb tenses are also considered[4] and changed accordingly. Other functions such as dropping adverbial clauses or changing tenses could be defined in a similar manner.

Both the knowledge-guided and hand-defined generators make local changes to the sentences based on simple rules. It should be possible to extend the hand-defined rules to cover the long tail (as long as they are procedurally definable). However, a more scalable approach would be to extend our generators to trainable models that can cover a wider range of phenomena than hand-defined rules. Moreover, the applicability of these rules generally depends on the context which can also be incorporated in such trainable generators.

3.3 Neural Generators

For each entailment class c, we use a trainable sequence-to-sequence neural model (Sutskever

[2] A similar approach was used in a parallel work to generate an adversarial dataset from SNLI (Glockner et al., 2018).

[3] Only 211 examples (2.11%) in the SNLI training set contain negation triggers such as not, 'nt, etc.

[4] https://www.nodebox.net/code/index.php/Linguistics

et al., 2014; Luong et al., 2015) to generate an entailment example (s, s', c) from an input sentence s. The seq2seq model, trained on examples labeled c, itself acts as the transformation function f_ρ of the corresponding generator $\mathbb{G}_c^{\text{s2s}}$. The label g_ρ is set to c. The joint probability of seq2seq model is:

$$\mathbb{G}_c^{\text{s2s}}(X_c; \phi_c) = \mathbb{G}_c^{\text{s2s}}(H_c, P_c; \phi_c) \quad (1)$$
$$= \Pi_i P(h_{i,c} | p_{i,c}; \phi_c) P(h_i) \quad (2)$$

The loss function for training the seq2seq is:

$$\hat{\phi}_c = \underset{\phi_c}{\text{argmin}}\, L(H_c, \mathbb{G}_c^{\text{s2s}}(X_c; \phi_c)) \quad (3)$$

where L is the cross-entropy loss between the original hypothesis H_c and the predicted hypothesis. Cross-entropy is computed for each predicted word w_i against the same in H_c given the sequence of previous words in H_c . $\hat{\phi}_c$ are the optimal parameters in $\mathbb{G}_c^{\text{s2s}}$ that minimize the loss for class c. We use the single most likely output to generate sentences in order to reduce decoding time.

3.4 Example Generation

The generators described above are used to create new entailment examples from the training data. For each example (p, h, c) in the data, we can create two new examples: $\big(p, f_\rho(p), g_\rho\big)$ and $\big(h, f_\rho(h), g_\rho\big)$.

The examples generated this way using $\mathbb{G}^{\text{KB}}$ and $\mathbb{G}^{\text{H}}$ can, however, be relatively easy, as the premise and hypothesis would differ by only a word or so. We therefore compose such simple ("first-order") generated examples with the original input example to create more challenging "second-order" examples. We can create second-order examples by composing the original example (p, h, c) with a generated sentence from hypothesis, $f_\rho(h)$ and premise, $f_\rho(p)$. Figure 1 depicts how these two kinds of examples are generated from an input example (p, h, c).

First, we consider the second-order example between the original premise and the transformed hypothesis: $(p, f_\rho(h), \bigoplus(c, g_\rho))$, where $\bigoplus$, defined in the left half of Table 3, composes the input example label c (connecting p and h) and the generated example label g_ρ to produce a new label. For instance, if p entails h and h entails $f_\rho(h)$, p would entail f_ρ. In other words, $\bigoplus(\sqsubseteq, \sqsubseteq)$ is $\sqsubseteq$. For example, composing ("A man is playing soccer", "A man is playing a game", $\sqsubseteq$) with a generated hypothesis $f_\rho(h)$: "A person is playing a game." will give a new second-order entailment example: ("A man is playing soccer", "A person is playing a game", $\sqsubseteq$).

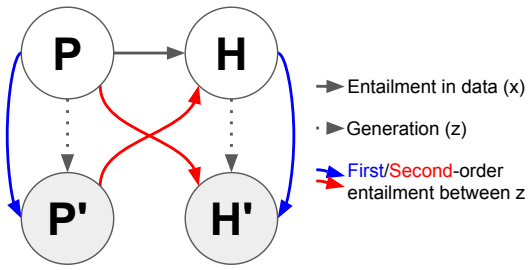

Figure 1: Generating first-order (blue) and second-order (red) examples.

$p \Rightarrow h$	$h \Rightarrow h'$	$p \Rightarrow h'$	$p \Rightarrow h$	$p \Rightarrow p'$	$p' \Rightarrow h$
c	g_ρ	$\bigoplus$	c	g_ρ	$\bigotimes$
$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	?
$\sqsubseteq$	人	人	$\sqsubseteq$	人	?
$\sqsubseteq$	#	#	$\sqsubseteq$	#	#
人	$\sqsubseteq$	?	人	$\sqsubseteq$	?
人	人	?	人	人	?
人	#	#	人	#	#
#	$\sqsubseteq$	#	#	$\sqsubseteq$	#
#	人	#	#	人	#
#	#	#	#	#	#

Table 3: Entailment label composition functions $\bigoplus$ (left) and $\bigotimes$ (right) for creating second-order examples. c and g_ρ are the original and generated labels, resp. $\sqsubseteq$: *entails*, 人: *contradicts*, #: *neutral*, ?: undefined

Second, we create an example from the generated premise to the original hypothesis: $(f_\rho(p), h, \bigotimes(g_\rho, c))$. The composition function here, denoted $\bigotimes$ and defined in the right half of Table 3, is often undetermined. For example, if p entails $f_\rho(p)$ and p entails h, the relation between $f_\rho(p)$ and h is undetermined i.e. $\bigotimes(\sqsubseteq, \sqsubseteq) = ?$. While this particular composition $\bigotimes$ often leads to undetermined or neutral relations, we use it here for completeness. For example, composing the previous example with a generated *neutral* premise, $f_\rho(p)$: "A person is wearing a cap" would generate an example ("A person is wearing a cap", "A man is playing a game", #)

The composition function $\bigoplus$ is the same as the "join" operation in natural logic reasoning (Icard III and Moss, 2014), except for two differences: (a) relations that do not belong to our

three entailment classes are mapped to '?', and (b) the exclusivity/alternation relation is mapped to *contradicts*. The composition function $\bigotimes$, on the other hand, does not map to the join operation.

3.5 Implementation Details

Given the original training examples X, we generate the examples from each premise and hypothesis in a batch using $\mathbb{G}^{\text{KB}}$ and $\mathbb{G}^{\text{H}}$. We also generate new hypothesis per class for each premise using $\mathbb{G}_c^{\text{s2s}}$. Using all the generated examples to train the model would, however, overwhelm the original training set. For examples, our knowledge-guided generators $\mathbb{G}^{\text{KB}}$ can be applied in 17,258,314 different ways.

To avoid this, we sub-sample our synthetic examples to ensure that they are proportional to the input examples X, specifically they are bounded to $\alpha|X|$ where α is tuned for each dataset. Also, as seen in Table 3, our knowledge-guided generators are more likely to generate *neutral* examples than any other class. To make sure that the labels are not skewed, we also sub-sample the examples to ensure that our generated examples have the same class distribution as the input batch. The SciTail dataset only contains two classes: *entails* mapped to $\sqsubseteq$ and *neutral* mapped to $\not\asymp$. As a result, generated examples that do not belong to these two classes are ignored.

The sub-sampling, however, has a negative side-effect where our generated examples end up using a small number of lexical relations from the large knowledge bases. On moderate datasets, this would cause the entailment model to potentially just memorize these few lexical relations. Hence, we generate new entailment examples for each mini-batch and update the model parameters based on the training+generated examples in this batch.

The overall example generation procedure goes as follows: For each mini-batch X (1) randomly choose 3 applicable rules per source and sentence (e.g., replacing men with people based on PPDB in premise is one rule), (2) produce examples Z_{all} using $\mathbb{G}^{\text{KB}}$, $\mathbb{G}^{\text{H}}$ and $\mathbb{G}^{\text{s2s}}$, (3) randomly sub-select examples Z from Z_{all} to ensure the balance between classes and $|Z| = \alpha|X|$.

4 AdvEntuRe

Figure 2 shows the complete architecture of our model, AdvEntuRe (ADVersarial training for textual ENTailment Using Rule-based Examples.). The entailment model $\mathbb{D}$ is shown with the white box and two proposed generators are shown using black boxes. We combine the two symbolic untrained generators, $\mathbb{G}^{\text{KB}}$ and $\mathbb{G}^{\text{H}}$ into a single $\mathbb{G}^{\text{rule}}$ model. We combine the generated adversarial examples Z with the original training examples X to train the discriminator. Next, we describe how the individual models are trained and finally present our new approach to train the generator based on the discriminator's performance.

4.1 Discriminator Training

We use one of the state-of-the-art entailment models (at the time of its publication) on SNLI, decomposable attention model (Parikh et al., 2016) with intra-sentence attention as our discriminator $\mathbb{D}$. The model attends each word in hypothesis with each word in the premise, compares each pair of the attentions, and then aggregates them as a final representation. This discriminator model can be easily replaced with any other entailment model without any other change to the AdvEntuRe architecture. We pre-train our discriminator $\mathbb{D}$ on the original dataset, X=(P, H, C) using:

$$\mathbb{D}(X;\theta) = \underset{\hat{C}}{\text{argmax}}\, \mathbb{D}(\hat{C}\,|\,P, H;\theta) \tag{4}$$

$$\hat{\theta} = \underset{\theta}{\text{argmin}}\, L(C, \mathbb{D}(X;\theta)) \tag{5}$$

where L is cross-entropy loss function between the true labels, Y and the predicted classes, and $\hat{\theta}$ are the learned parameters.

4.2 Generator Training

Our knowledge-guided and hand-defined generators are symbolic parameter-less methods which are not currently trained. For simplicity, we will refer to the set of symbolic rule-based generators as $\mathbb{G}^{\text{rule}} := \mathbb{G}^{\text{KB}} \cup \mathbb{G}^{\text{H}}$. The neural generator $\mathbb{G}^{\text{s2s}}$, on the other hand, can be trained as described earlier. We leave the training of the symbolic models for future work.

4.3 Adversarial Training

We now present our approach to iteratively train the discriminator and generator in a GAN-style framework. Unlike traditional GAN (Goodfellow et al., 2014) on image/text generation that aims to obtain better generators, our goal is to build a robust discriminator regularized by the generators ($\mathbb{G}^{\text{s2s}}$ and $\mathbb{G}^{\text{rule}}$). The discriminator and generator are iteratively trained against each other to

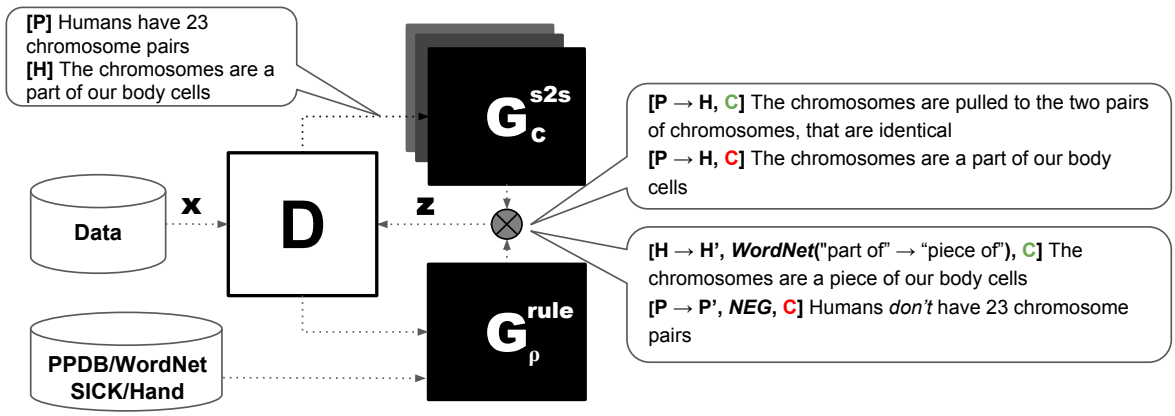

Figure 2: Overview of AdvEntuRe, our model for knowledge-guided textual entailment.

Algorithm 1 Training procedure for AdvEntuRe.

```
pretrain discriminator 𝔻(θ̂) on X;
pretrain generators 𝔾_c^s2s(φ̂) on X;
for number of training iterations do
  for mini-batch B ← X do
    generate examples from 𝔾
      Z_G ⇐ 𝔾(B; φ),
    balance X and Z_G s.t. |Z_G| ≤ α|X|
    optimize discriminator:
      θ̂ = argmin_θ L_𝔻(X + Z_G; θ)
    optimize generator:
      φ̂ = argmin_φ L_{𝔾^s2s}(Z_G; L_𝔻; φ)
    Update θ ← θ̂; φ ← φ̂
```

achieve better discrimination on the augmented data from the generator and better example generation against the learned discriminator. Algorithm 1 shows our training procedure.

First, we pre-train the discriminator $\mathbb{D}$ and the seq2seq generators $\mathbb{G}^{s2s}$ on the original data X. We alternate the training of the discriminator and generators over K iterations (set to 30 in our experiments).

For each iteration, we take a mini-batch B from our original data X. For each mini-batch, we generate new entailment examples, Z_G using our adversarial examples generator. Once we collect all the generated examples, we balance the examples based on their source and label (as described in Section 3.5). In each training iteration, we optimize the discriminator against the augmented training data, $X + Z_G$ and use the discriminator loss to guide the generator to pick challenging examples. For every mini-batch of examples $X + Z_G$, we compute the discriminator loss $L(C; \mathbb{D}(X + Z_G; \theta))$ and apply the negative of this loss to each word of the generated sentence in $\mathbb{G}^{s2s}$. In other words, the discriminator loss value replaces the cross-entropy loss used to train the seq2seq model (similar to a REINFORCE (Williams, 1992) reward). This basic approach uses the loss over the entire batch to update the generator, ignoring whether specific examples were hard or easy for the discriminator. Instead, one could update the generator per example based on the discriminator's loss on that example. We leave this for future work.

5 Experiments

Our empirical assessment focuses on two key questions: (a) Can a handful of rule templates improve a state-of-the-art entailment system, especially with moderate amounts of training data? (b) Can iterative GAN-style training lead to an improved discriminator?

To this end, we assess various models on the two entailment **datasets** mentioned earlier: SNLI (570K examples) and SciTail (27K examples).[5] To test our hypothesis that adversarial example based training prevents overfitting in small to moderate training data regimes, we compare model accuracies on the test sets when using 1%, 10%, 50%, and 100% subsamples of the train and dev sets.

We consider two baseline **models**: $\mathbb{D}$, the Decomposable Attention model (Parikh et al., 2016) with intra-sentence attention using pre-trained word embeddings (Pennington et al., 2014); and $\mathbb{D}_{\text{retro}}$ which extends $\mathbb{D}$ with word embeddings initialized by retrofitted vectors (Faruqui et al., 2015). The vectors are retrofitted on PPDB, Word-

[5] SNLI has a 96.4%/1.7%/1.7% split and SciTail has a 87.3%/4.8%/7.8% split on train, valid, and test sets, resp.

Table 4: Test accuracies with different subsampling ratios on SNLI (top) and SciTail (bottom).

SNLI	1%	10%	50%	100%
$\mathbb{D}$	57.68	75.03	82.77	84.52
$\mathbb{D}_{\text{retro}}$	57.04	73.45	81.18	84.14
ADVENTURE				
⌞ $\mathbb{D}$ + $\mathbb{G}^{\text{s2s}}$	58.35	75.66	82.91	**84.68**
⌞ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$	**60.45**	**77.11**	**83.51**	84.40
⌞ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ + $\mathbb{G}^{\text{s2s}}$	59.33	76.03	83.02	83.25

SciTail	1%	10%	50%	100%
$\mathbb{D}$	56.60	60.84	73.24	74.29
$\mathbb{D}_{\text{retro}}$	59.75	67.99	69.05	72.63
ADVENTURE				
⌞ $\mathbb{D}$ + $\mathbb{G}^{\text{s2s}}$	**65.78**	**70.77**	74.68	76.92
⌞ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$	61.74	66.53	73.99	**79.03**
⌞ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ + $\mathbb{G}^{\text{s2s}}$	63.28	66.78	**74.77**	78.60

Net, FrameNet, and all of these, with the best results for each dataset reported here.

Our proposed model, ADVENTURE, is evaluated in three flavors: $\mathbb{D}$ augmented with examples generated by $\mathbb{G}^{\text{rule}}$, $\mathbb{G}^{\text{s2s}}$, or both, where $\mathbb{G}^{\text{rule}} = \mathbb{G}^{\text{KB}} \cup \mathbb{G}^{\text{H}}$. In the first two cases, we create new examples for each batch in every epoch using a fixed generator (cf. Section 3.5). In the third case ($\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ + $\mathbb{G}^{\text{s2s}}$), we use the GAN-style training.

We uses grid search to find the best hyperparameters for $\mathbb{D}$ based on the validation set: hidden size 200 for LSTM layer, embedding size 300, dropout ratio 0.2, and fine-tuned embeddings.

The ratio between the number of generated vs. original examples, α is empirically chosen to be 1.0 for SNLI and 0.5 for SciTail, based on validation set performance. Generally, very few generated examples (small α) has little impact, while too many of them overwhelm the original dataset resulting in worse scores (cf. Appendix for more details).

5.1 Main Results

Table 4 summarizes the test set accuracies of the different models using various subsampling ratios for SNLI and SciTail training data.

We make a few observations. First, $\mathbb{D}_{\text{retro}}$ is ineffective or even detrimental in most cases, except on SciTail when 1% (235 examples) or 10% (2.3K examples) of the training data is used. The gain in these two cases is likely because retrofitted lexical rules are helpful with extremely less data training while not as data size increases.

On the other hand, our method always achieves the best result compared to the baselines ($\mathbb{D}$ and $\mathbb{D}_{\text{retro}}$). Especially, significant improvements are made in less data setting: +2.77% in SNLI (1%) and 9.18% in SciTail (1%). Moreover, $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$'s accuracy on SciTail (100%) also outperforms the previous state-of-the-art model (DGEM (Khot et al., 2018), which achieves 77.3%) for that dataset by 1.7%.

Table 5: Test accuracies across various rules $\mathcal{R}$ and classes $\mathcal{C}$. Since SciTail has two classes, we only report results on two classes of $\mathbb{G}^{\text{s2s}}$

	$\mathcal{R}/\mathcal{C}$	SNLI (5%)	SciTail (10%)
$\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$	$\mathbb{D}$	69.18	60.84
	+ PPDB	**72.81 (+3.6%)**	65.52 (+4.6%)
	+ SICK	71.32 (+2.1%)	67.49 (+6.5%)
	+ WordNet	71.54 (+2.3%)	64.67 (+3.8%)
	+ HAND	71.15 (+1.9%)	**69.05 (+8.2%)**
	+ all	71.31 (+2.1%)	64.16 (+3.3%)
$\mathbb{D}$ + $\mathbb{G}^{\text{s2s}}$	$\mathbb{D}$	69.18	60.84
	+ positive	71.21 (+2.0%)	67.49 (+6.6%)
	+ negative	71.76 (+2.6%)	68.95 (+8.1%)
	+ neutral	71.72 (+2.5%)	-
	+ all	**72.28 (+3.1%)**	**70.77 (+9.9%)**

Among the three different generators combined with $\mathbb{D}$, both $\mathbb{G}^{\text{rule}}$ and $\mathbb{G}^{\text{s2s}}$ are useful in SciTail, while $\mathbb{G}^{\text{rule}}$ is much more useful than $\mathbb{G}^{\text{s2s}}$ on SNLI. We hypothesize that seq2seq model trained on large training sets such as SNLI will be able to reproduce the input sentences. Adversarial examples from such a model are not useful since the entailment model uses the same training examples. However, on smaller sets, the seq2seq model would introduce noise that can improve the robustness of the model.

5.2 Ablation Study

To evaluate the impact of each generator, we perform ablation tests against each symbolic generator in $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ and the generator $\mathbb{G}^{\text{s2s}}_c$ for each entailment class c. We use a 5% sample of SNLI and a 10% sample of SciTail. The results are summarized in Table 5.

Interestingly, while PPDB (phrasal paraphrases) helps the most (**+3.6%**) on SNLI, simple negation rules help significantly (**+8.2%**) on SciTail dataset. Since most entailment examples in SNLI are minor rewrites by Turkers, PPDB often contains these simple paraphrases. For SciTail, the sentences are authored independently with limited gains from simple paraphrasing. However, a model trained on only 10% of the dataset (2.3K

Table 6: Given a premise **P** (underlined), examples of hypothesis sentences **H'** generated by seq2seq generators $\mathbb{G}^{\text{s2s}}$, and premise sentences **P'** generated by rule based generators $\mathbb{G}^{\text{rule}}$, on the full SNLI data. Replaced words or phrases are shown in **bold**. This illustrates that even simple, easy-to-define rules can generate useful adversarial examples.

P	a person on a horse jumps over a broken down airplane
H': $\mathbb{G}^{\text{s2s}}_{c=\sqsubseteq}$	a person is on a horse jumps over a rail, a person jumping over a plane
H': $\mathbb{G}^{\text{s2s}}_{c=人}$	a person is riding a horse in a field with a dog in a red coat
H': $\mathbb{G}^{\text{s2s}}_{c=\#}$	a person is in a blue dog is in a park
P (or **H**)	a dirt bike rider catches some air going off a large hill
P': $\mathbb{G}^{\text{KB(PPDB)}}_{\rho=\equiv, g_\rho=\sqsubseteq}$	a dirt **motorcycle** rider catches some air going off a large hill
P': $\mathbb{G}^{\text{KB(SICK)}}_{\rho=c, g_\rho=\#}$	a dirt bike **man on yellow bike** catches some air going off a large hill
P': $\mathbb{G}^{\text{KB(WordNet)}}_{\rho=syno, g_\rho=\sqsubseteq}$	a dirt bike rider catches some **atmosphere** going off a large hill
P': $\mathbb{G}^{\text{Hand}}_{\rho=\text{NEG}, g_\rho=人}$	a dirt bike rider **do not catch** some air going off a large hill

examples) would end up learning a model relying on purely word overlap. We believe that the simple negation examples introduce *neutral* examples with high lexical overlap, forcing the model to find a more informative signal.

On the other hand, using all classes for $\mathbb{G}^{\text{s2s}}$ results in the best performance, supporting the effectiveness of the GAN framework for penalizing or rewarding generated sentences based on $\mathbb{D}$'s loss. Preferential selection of rules within the GAN framework remains a promising direction.

5.3 Qualitative Results

Table 6 shows examples generated by various methods in AdvEntuRe. As shown, both seq2seq and rule based generators produce reasonable sentences according to classes and rules. As expected, seq2seq models trained on very few examples generate noisy sentences. The quality of our knowledge-guided generators, on the other hand, does not depend on the training set size and they still produce reliable sentences.

5.4 Case Study: Negation

For further analysis of the negation-based generator in Table 1, we collect only the negation examples in test set of SNLI, henceforth referred to as nega-SNLI. Specifically, we extract examples where either the premise or the hypothesis contains "not", "no", "never", or a word that ends with "n't'. These do not cover more subtle ways of expressing negation such as "seldom" and the use of antonyms. nega-SNLI contains 201 examples with the following label distribution: 51 (25.4%) neutral, 42 (20.9%) entails, 108 (53.7%) contradicts. Table 7 shows examples in each category.

Table 7: Negation examples in nega-SNLI

$\sqsubseteq$	P: several women are playing volleyball. H: this doesn't look like soccer.
#	P: a man with no shirt on is performing with a baton. H: a man is trying his best at the national championship of baton.
人	P: island native fishermen reeling in their nets after a long day's work. H: the men did not go to work today but instead played bridge.

While $\mathbb{D}$ achieves an accuracy of only 76.64%[6] on nega-SNLI, $\mathbb{D}$ + $\mathbb{G}^{\text{H}}$ with NEGATE is substantially more successful (+6.1%) at handling negation, achieving an accuracy of 82.74%.

6 Conclusion

We introduced an adversarial training architecture for textual entailment. Our seq2seq and knowledge-guided example generators, trained in an end-to-end fashion, can be used to make any base entailment model more robust. The effectiveness of this approach is demonstrated by the significant improvement it achieves on both SNLI and SciTail, especially in the low to medium data regimes. Our rule-based generators can be expanded to cover more patterns and phenomena, and the seq2seq generator extended to incorporate per-example loss for adversarial training.

[6]This is much less than the full test accuracy of 84.52%.

References

Gabor Angeli and Christopher D Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *EMNLP*, pages 534–545.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42:763–808.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2018. Natural language inference with external knowledge. In *ACL*.

LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. 2017. Triple generative adversarial nets. In *NIPS*, pages 4091–4101.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. *NAACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL-HLT*, pages 758–764.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *ACL*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. *ICLR*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*, pages 2672–2680.

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

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *EMNLP*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *ACL*.

Thomas Icard III and Lawrence Moss. 2014. Recent progress in monotonicity. *LiLT (Linguistic Issues in Language Technology)*, 9.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke S. Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*.

R. Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and explaining causes from text for a time series event. In *EMNLP*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. *AAAI*.

George Lakoff. 1970. Linguistics and Natural Logic. *Synthese*, 22(1-2):151–271.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Bill MacCartney and Christopher D. Manning. 2012. Natural logic and natural language inference. In *Computing Meaning. Text, Speech and Language Technology*, volume 47.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

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

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *HLT-NAACL*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher D Manning, and Andrew Y Ng. 2005. Robust textual inference using diverse knowledge sources. In *1st PASCAL Recognition Textual Entailment Challenge Workshop*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Recognizing textual entailment via multi-task knowledge assisted lstm. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 285–298. Springer.

Vivian S Silva, André Freitas, and Siegfried Handschuh. 2018. Recognizing and justifying text entailment through distributional navigation on definition graphs. In *AAAI*.

Fred Sommers. 1982. The logic of natural language.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Shanshan Wang and Lei Zhang. 2017. CatGAN: Coupled adversarial transfer for domain generation. *CoRR*, abs/1711.08904.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.