Make Up Your Mind! Adversarial Generation of Inconsistent Natural Language Explanations

Oana-Maria Camburu[1] **Brendan Shillingford**[1,2] **Pasquale Minervini**[3]
Thomas Lukasiewicz[1,4] **Phil Blunsom**[1,2]
[1]University of Oxford [2]DeepMind, London
[3]University College London [4]Alan Turing Institute, London
firstname.lastname@cs.ox.ac.uk p.minervini@ucl.ac.uk

Abstract

To increase trust in artificial intelligence systems, a growing amount of works are enhancing these systems with the capability of producing natural language explanations that support their predictions. In this work, we show that such appealing frameworks are nonetheless prone to generating inconsistent explanations, such as "A dog is an animal" and "A dog is not an animal", which are likely to decrease users' trust in these systems. To detect such inconsistencies, we introduce a simple but effective adversarial framework for generating a *complete* target sequence, a scenario that has not been addressed so far. Finally, we apply our framework to a state-of-the-art neural model that provides natural language explanations on SNLI, and we show that this model is capable of generating a significant amount of inconsistencies.

1 Introduction

For machine learning systems to be widely adopted in practice, they need to be trusted by users [10]. However, the black-box nature of neural networks can create doubt or lack of trust, especially since recent works show that highly accurate models can heavily rely on annotation artifacts [13, 6]. In order to increase users' trust in these systems, a growing number of works [4, 26, 19, 14, 21] enhance neural networks with an explanation generation module that is jointly trained to produce natural language explanations for their final decisions. The supervision on the explanations usually comes from human-provided explanations for the ground-truth answers.

In this work, we first draw attention to the fact that the explanation module may generate inconsistent explanations. For example, a system that generates "Snow implies outdoors" for justifying one prediction, and "Snow implies indoors" for justifying another prediction would likely decrease users' trust in the system. We note that, while users may already decrease their trust in a model that generates incorrect statements, such as "Snow implies indoors", if these statements are consistent over the input space, the users might, at least, be reassured that the explanations are a good reflection of the inner workings of the model. Subsequently, they may not trust the model when it is applied on certain concepts, such as the snow's location, but they may trust the model on other concepts where it has shown a persistently correct understanding.

Generating Adversarial Explanations. Adversarial examples [29] are inputs that have been specifically designed by an adversary to cause a machine learning algorithm to produce an incorrect answer [2]. In this work, we focus on the problem of *generating adversarial explanations*. More specifically, given a machine learning model that can jointly produce predictions and their explanations, we propose a framework that can identify inputs that cause the model to generate *mutually inconsistent explanations*.

NeurIPS 2019 Workshop on Safety and Robustness in Decision Making, Vancouver, Canada

To this date, most of the research on adversarial examples in computer vision focuses on generating adversarial perturbations that are imperceptible to humans, but make the machine learning model to produce a different prediction [12]. Similarly, in natural language processing, most of the literature focuses on identifying semantically invariant modifications of natural language sentences that cause neural models to change their predictions [32].

Our problem has three desired properties that make it different from commonly researched adversarial setups:

1. The model has to generate a *complete* target sequence, i.e., the attack is considered successful if the model generates an explanation that is inconsistent with a given generated explanation. This is more challenging than the adversarial setting commonly addressed in sequence-to-sequence models, where the objective is generating sequences characterized by the presence or absence of certain given tokens [8, 33].
2. The adversarial input does not have to be a paraphrase or a small perturbation of the original input, since our objective is generating mutually inconsistent explanations and not a label attack. [1]
3. We strongly prefer the adversarial inputs to be grammatically correct English sentences — in previous works, this requirement never appears jointly with the aforementioned two requirements.

To our knowledge, our work is the first to tackle this problem setting, especially due to the complete target requirement, which is a challenging requirement for sequence generation. The simple yet effective framework that we introduce for the above scenario consists of training a neural network, which we call REVERSEJUSTIFIER, to invert the explanation module, i.e., to find an input for which the model will produce a given explanation. We further create simple rules to construct a set of potentially inconsistent explanations, and query the REVERSEJUSTIFIER model for inputs that could lead the original model to generate these adversarial explanations. When applied to the best explanation model from Camburu et al. [4], our procedure detects an estimated 460 distinct pairs of inconsistencies on the e-SNLI test set.

2 The e-SNLI Dataset

The natural language inference task consists in detecting whether a pair of sentences, called premise and hypothesis, are in a relation of: *entailment*, if the premise entails the hypothesis; *contradiction*, if the premise contradicts the hypothesis; or *neutral*, if neither entailment nor contradiction holds. The SNLI corpus [3] of $\sim$570K such human-written instances enabled a plethora of works on this task [28, 25, 22]. Recently, Camburu et al. [4] augmented SNLI with crowd-sourced free-form explanations of the ground-truth label, called e-SNLI. Their best model for generating explanations, called EXPLAINTHENPREDICTATTENTION (hereafter called ETPA), is a sequence-to-sequence attention model that uses two bidirectional LSTM networks [16] for encoding the premise and hypothesis, and an LSTM decoder for generating the explanation while separately attending over the tokens of the premise and hypothesis. Furthermore, they predict the label solely based on the explanation via a separately trained neural network, which maps an explanation to a label. In our work, we show that our simple attack on the explanation generation network is able to detect a significant amount of inconsistent explanation generated by ETPA. We highlight that our final goal is not the label attack, even if for this particular model, since the label is predicted solely from the explanation, we implicitly also have a label attack with high probability.[2]

3 Method

We define two explanations to be *inconsistent* if they provide logically contradictory arguments. For example, "Seagulls are birds." and "Seagulls and birds are different animals." [3] are inconsistent explanations. Our baseline method consists of the following 5 steps:

[1] Ideally, the explanation and predicted label align, but in general it may not be the case.

[2] The explanation-to-label model had a test accuracy of 96.83%.

[3] This is a real example of an inconsistency detected by our method.

1. Reverse the explanation module by training a REVERSEJUSTIFIER model to map from a generated explanation to an input that causes the model to generate this explanation.
2. For each originally generated explanation by the ETPA, generate a list of statements that are inconsistent with the this explanation — we call them *adversarial explanations*.
3. Query the REVERSEJUSTIFIER model on each adversarial explanation to get what we will call *reverse inputs* — i.e., inputs that may cause the model to produce adversarial explanations.
4. Feed the reverse inputs into the original model to get the *reverse explanations*.
5. Check if any of the reverse explanations are indeed inconsistent with the original one.

In the following, we detail how we instantiate our procedure on e-SNLI.

4 Experiments

In this work, we use the trained ETPA model[4] from Camburu et al. [4], which gave the highest percentage of correct explanations (64.7%). In our experiments, for the REVERSEJUSTIFIER model, we use the same neural network architecture and hyperparameters used by Camburu et al. [4] for their attention model, with the difference that inputs are now premise-explanation pairs rather than premise-hypothesis pairs, and outputs are hypotheses rather than explanations. Given a premise and an explanation, our REVERSEJUSTIFIER model is able to reconstruct the correct hypothesis 32.78% of the times on the e-SNLI test set. We found it satisfactory to reverse only the hypothesis; however, it is possible to jointly reverse both premise and hypothesis, which may result in detecting more inconsistencies due to the exploration of a larger portion of the input space.

To perform Step 2, we note that the explanations in e-SNLI naturally follow label-specific templates. For example, annotators often used "One cannot *[X]* and *[Y]* simultaneously" to justify a contradiction, "Just because *[X]*, doesn't mean *[Y]*" for neutral, or "*[X]* implies *[Y]*" for entailment. Since two labels are mutually exclusive, transforming an explanation from one template to a template of another label automatically creates an inconsistency. For example, for the explanation of the contradiction "One cannot eat and sleep simultaneously", we match *[X]*="eat" and *[Y]*="sleep", and we create the inconsistent explanation "Eat implies sleep" using the entailment template "*[X]* implies *[Y]*". We note that this type of rule-based procedure is not applicable only to e-SNLI. Since explanations are by nature logical sentences, for any task, one may define a set of rules that the explanations should adhere to. For example, for explanations in self-driving cars [19], one can interchange "green light" with "red light", or "stop" with "accelerate", to get inconsistent — and potentially hazardous! — explanations such as "The car accelerates, because it is red light". Similarly, in law applications, one can interchange "guilty" with "innocent", or "arrest" with "release". Therefore, our rule-based generation strategy — and the whole framework — can be applied to any task where one is required to test its explanations against an essential set of predefined task-specific inconsistencies, and our paper encourages the community to consider such hazardous inconsistencies for their tasks.

To summarize, on e-SNLI, we first created, for each label, a list of the most used templates that we manually identified by inspecting the human annotated explanations. We provide the lists of templates in Appendix A.1. We then proceeded as follows: for each explanation generated by ETPA on the SNLI test set, we first reversed negations (if applicable) by simply removing the "not" and "n't" tokens.[5] Secondly, we tried to match the explanation to a template. If there was no negation and no template match, we discarded the instance. We only discarded 2.6% of the SNLI test set in this way. If a template was found, we identified its associated label L and retrieved the matched substrings *[X]* and *[Y]*. For each of the templates associated with the two other labels different from L, we substituted *[X]* and *[Y]* with the corresponding strings. We note that this procedure may result in grammatically or semantically incorrect adversarial explanations, especially since we did not perform any linguistic-specific adjustments. However, our REVERSEJUSTIFIER turned out to perform well in smoothing out these errors and in generating grammatically correct reverse hypotheses. This is not surprising, since it has been trained to output the ground-truth correct hypothesis. Specifically, we manually annotated 100 random instances of reversed hypotheses generated by REVERSEJUSTIFIER and found 81 to be both grammatically and semantically valid sentences.

[4] From: `https://github.com/OanaMariaCamburu/e-SNLI`

[5] During pre-processing, the tokenizer splits words such as "don't" into two tokens: "do" and "n't".

Table 1: Examples of three true detected inconsistencies (1)–(3) and one false detected inconsistency (4).

(1) PREMISE: A guy in a red jacket is snowboarding in midair.	
(a) ORIGINAL HYPOTHESIS: A guy is outside in the snow. PREDICTED LABEL: entailment ORIGINAL EXPLANATION: **Snowboarding is done outside.**	(b) REVERSE HYPOTHESIS: The guy is outside. PREDICTED LABEL: contradiction REVERSE EXPLANATION: **Snowboarding is not done outside.**
(2) PREMISE: A man talks to two guards as he holds a drink.	
(a) ORIGINAL HYPOTHESIS: The prisoner is talking to two guards in the prison cafeteria. PREDICTED LABEL: neutral ORIGINAL EXPLANATION: **The man is not necessarily a prisoner.**	(b) REVERSE HYPOTHESIS: A prisoner talks to two guards. PREDICTED LABEL: entailment REVERSE EXPLANATION: **A man is a prisoner.**
(3) PREMISE: A woman in a black outfit lies face first on a yoga mat; several paintings are hanged on the wall, and the sun shines through a large window near her.	
(a) ORIGINAL HYPOTHESIS: There is a person in a room. PREDICTED LABEL: contradiction ORIGINAL EXPLANATION: **A woman is not a person.**	(b) REVERSE HYPOTHESIS: A person is on a yoga mat. PREDICTED LABEL: entailment REVERSE EXPLANATION: **A woman is a person.**
(4) PREMISE: A female acrobat with long, blond curly hair, dangling upside down while suspending herself from long, red ribbons of fabric.	
(a) ORIGINAL HYPOTHESIS: A horse jumps over a fence. PREDICTED LABEL: contradiction ORIGINAL EXPLANATION: A female is not a horse.	(b) REVERSE HYPOTHESIS: The female has a horse. PREDICTED LABEL: neutral REVERSE EXPLANATION: Not all female have a horse.

For each adversarial explanation, we queried the REVERSEJUSTIFIER module and subsequently fed each obtained reverse hypothesis back to the ETPA model to get the reverse explanation. To check whether the reverse explanation was inconsistent with the original one, we again used the list of adversarial explanations generated at Step 2 and checked for an exact string match. It is likely that, at this step, we discarded a large amount of inconsistencies, due to insignificant syntactic differences. However, when an exact match was found, i.e., a *potential inconsistency*, it is very likely to be a *true inconsistency*. Indeed, we manually annotated a random sample of 100 pairs of potential inconsistencies and found 85% to be true inconsistencies.

More precisely, our procedure first identified a total of 1045 pairs of potential inconsistencies for the ETPA model applied on the test set of e-SNLI. However, multiple distinct reverse hypotheses gave rise to the same reverse explanation. On average, we found that there are 1.93 ± 1.77 distinct reverse hypotheses giving rise to the same reverse explanation. Therefore, we counted a total of 541 distinct pairs of potentially inconsistent explanations. Given our estimation of 85% to be true inconsistencies, we obtained a total of ≈ 460 distinct true inconsistencies. While this means that our procedure only has a success rate of 4.68%, it is nonetheless alarming that this very simple, under-optimized framework detects a significant amount of inconsistencies on a model trained on $\sim$570K instances.

In Table 1, we can see three examples of true inconsistencies detected by our procedure and one example of a false inconsistency. In Example (3), we notice that the incorrect explanation was actually given on the original hypothesis.

Manual Scanning. Finally, we were curious to what extent a simple manual scanning would find inconsistent explanations in the e-SNLI test set alone. We performed two such experiments. First, we manually analyzed the first 50 instances in the test set without finding any inconsistency. However, these examples were involving different concepts, thus decreasing the likelihood of finding inconsistencies. To account for this, in our second experiment, we constructed three groups around the concepts of *woman*, *prisoner*, and *snowboarding*, by simply selecting the explanations in the test set containing these words. We selected these concepts, because our framework detected inconsistencies about them — examples are listed in Table 1.

For *woman*, we obtained 1150 examples, and we looked at a random sample of 20 among which we did not find any inconsistency. For *snowboarding*, we found 16 examples in the test set and again no inconsistency among them. For *prisoner*, we only found one instance in the test set, so we had no ways to find out that the model is inconsistent with respect to this concept simply by scanning the test set.

We only looked at the test set for a fair comparison with our method that was only applied on this set. However, we highlight that manual scanning should not be regarded as a proper baseline, since it does not bring the same benefits as our framework. Indeed, manual scanning requires considerable human effort to look over a large set of explanations and find if any two are inconsistent.[6] Moreover,

[6]Even a group of only 50 explanations required non-negligible time.

restricting ourselves to the instances in the original dataset would clearly be less effective than being able to generate new instances from the input distribution. Our framework addresses these issues and provides direct pairs of very likely (approx. 85%) inconsistent explanations. Nonetheless, we considered this experiment useful for illustrating that the explanation module does not provide inconsistent explanations in a frequent manner. In fact, during our scanning over explanations, we also experimented with a few manually created potential adversarial hypothesis from Carmona et al. [5]. We were pleased to notice a good level of robustness against inconsistencies. For example, for the neutral pair (premise: "A bird is above water.", hypothesis: "A swan is above water."), we get the explanation "Not all birds are a swan.", while when interchanging bird with swan (premise: "A swan is above water.", hypothesis: "A bird is above water."), ETPA states that "A swan is a bird." Similarly, interchanging "child" with "toddler" in (premise: "A small child watches the outside world through a window.", hypothesis: "A small toddler watches the outside world through a window.") does not confuse the networks, which outputs "Not every child is a toddler." and "A toddler is a small child.", respectively. Further investigation on whether the networks can be tricked on concepts where it seems to exhibit robustness, such as *toddler* or *swan*, are left for future work.

5 Related Work

Explanatory Methods. Explaining predictions made by complex machine learning systems has been of increasing concern [9]. These explanations can be divided into two categories: feature importance explanations and full-sentence natural language explanations. The methods that provide feature importance explanations [27, 23, 7, 20, 11] aim to provide the user with the subset of input tokens that contributed the most to the prediction of the model. As pointed out by Camburu et al. [4], these explanations are not comprehensive, as one would need to infer the missing links between the words in order to form a complete argument. For example, in the natural language inference task, if the explanation is formed by the words "dog" and "animal", one would not know if the model learned that "A dog is an animal" or "An animal is a dog" or maybe even that "Dog and animal implies entailment". It is also arguably more user-friendly to get a full sentence explanation rather than a set of tokens. Therefore, an increasing amount of works focus on providing full sentence explanations [4, 19, 14]. However, generating fluent argumentation, while more appealing, it is also arguably a harder and more risky task. For example, similar in spirit to our work, Hendricks et al. [15] identified the risk of mentioning attributes from a strong class prior without any evidence being present in the input. In our work, we bring awareness to the risk of generating inconsistent explanations.

Generating Adversarial Examples. Generating adversarial examples has received increasing attention in natural language processing [31, 30]. However, most works in this space build on the requirement that the adversarial input should be a small perturbation [1, 17] or be preserving the main semantics [18] of the original input, but leading to a different prediction. While this is necessary for testing the stability of a model, our goal does not require the adversarial input to be semantically equivalent to the original, and any pair of correct English inputs that causes the model to produce inconsistent explanations suffices. On the other hand, the aforementioned models do not always require the adversarial input to be grammatically correct, and often they can change words or characters to completely random ones [8]. This assumption is acceptable for certain use cases, such as summarization of long pieces of text, where changing a few words would likely not change the main flow of the text. However, in our case, the inputs are short sentences and the model is being tested for robustness in fine-grained reasoning and common-sense knowledge, therefore it is more desirable to test the model on grammatically correct sentences.

Most importantly, to our knowledge, no previous adversarial attack for sequence-to-sequence models produces a *complete* target sequence. The closest to this goal, Cheng et al. [8] requires the presence of certain tokens anywhere in the target sequence. They only test with up to 3 required tokens, and their success rate dramatically drops from 99% for 1 required token to 37% for 3 tokens for the task of summarization. Similarly, Zhao et al. [33] proposed an adversarial framework for obtaining only the presence or absence of certain tokens in the target sequence for the task of machine translation. Our scenario would require as many tokens as the desired adversarial explanation, and we also additionally need them to be in a given order, thus tackling a much challenging task.

Finally, Minervini and Riedel [24] attempted to find inputs where a model trained on SNLI violates a set of logical constraints. This scenario may in theory lead to also finding inputs that lead to inconsistent explanations. However, their method needs to enumerate and evaluate a potentially very large set of perturbations of the inputs, obtained by, e.g., removing sub-trees or replacing tokens with their synonyms. While they succeed in finding adversarial examples, finding exact inconsistent explanations is a harder task, and hence their approach would be significantly more computationally challenging. Additionally, their perturbations are rule-based, and hence can easily generate incorrect English text. Moreover, their scenario does not require addressing the question of automatically producing undesired — in our case *inconsistent* — sequences.

Therefore, our work introduces a new practical attack scenario, and proposes a simple yet effective procedure, which we hope will be further improved by the community.

6 Summary and Outlook

In this work, we identified an essential shortcoming of the class of models that produce natural language explanations for their own decisions: the fact that such models are prone to producing inconsistent explanations, which can undermine users' trust in the model. We introduced a framework for identifying pairs of inconsistent explanations. We instantiated our procedure on the best explanation model available in the literature on e-SNLI, and obtained a significant amount of inconsistencies generated by this model.

The concern that we raise is general and can have a large practical impact. For example, humans would likely not accept a self-driving car if its explanation module — for example, the one proposed by Kim et al. [19] — is prone to state that "The car accelerates, because there is a red light at the intersection".

Future work will focus on two directions: developing more advanced procedures for detecting inconsistencies, and preventing the explanation modules from generating such inconsistencies.

Acknowledgments This work was supported by JP Morgan PhD Fellowship 2019-2020 and by the Alan Turing Institute under the EPSRC grant EP/N510129/1, and EPSRC grant EP/R013667/1.

References

[1] Belinkov, Y. and Bisk, Y. (2017). Synthetic and natural noise both break neural machine translation. *CoRR*, abs/1711.02173.

[2] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer.

[3] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326.

[4] Camburu, O., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. (2018). e-SNLI: Natural language inference with natural language explanations. In *NeurIPS*, pages 9560–9572.

[5] Carmona, V. I. S., Mitchell, J., and Riedel, S. (2018). Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *NAACL-HLT*, pages 1975–1985. Association for Computational Linguistics.

[6] Chen, D., Bolton, J., and Manning, C. D. (2016). A thorough examination of the cnn/daily mail reading comprehension task. *CoRR*, abs/1606.02858.

[7] Chen, J., Song, L., Wainwright, M., and Jordan, M. (2018). Learning to explain: An information-theoretic perspective on model interpretation. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 883–892, Stockholm, Sweden. PMLR.

[8] Cheng, M., Yi, J., Zhang, H., Chen, P., and Hsieh, C. (2018). Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128.

[9] Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv*.

[10] Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., and Beck, H. P. (2003). The role of trust in automation reliance. *Int. J. Hum.-Comput. Stud.*, 58(6):697–718.

[11] Feng, S., Wallace, E., II, A. G., Iyyer, M., Rodriguez, P., and Boyd-Graber, J. L. (2018). Pathologies of neural models make interpretation difficult. In *EMNLP*, pages 3719–3728. Association for Computational Linguistics.

[12] Goodfellow, I. J., McDaniel, P. D., and Papernot, N. (2018). Making machine learning robust against adversarial inputs. *Commun. ACM*, 61(7):56–66.

[13] Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. (2018). Annotation artifacts in natural language inference data. In *Proc. of NAACL*.

[14] Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., and Darrell, T. (2016). Generating visual explanations. In *ECCV (4)*, volume 9908 of *LNCS*, pages 3–19. Springer.

[15] Hendricks, L. A., Hu, R., Darrell, T., and Akata, Z. (2017). Grounding visual explanations (extended abstract). *CoRR*, abs/1711.06465.

[16] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[17] Hosseini, H., Xiao, B., and Poovendran, R. (2017). Deceiving Google's cloud video intelligence API built for summarizing videos. In *CVPR Workshops*, pages 1305–1309. IEEE Computer Society.

[18] Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. *CoRR*, abs/1804.06059.

[19] Kim, J., Rohrbach, A., Darrell, T., Canny, J. F., and Akata, Z. (2018). Textual explanations for self-driving vehicles. In *ECCV (2)*, volume 11206 of *Lecture Notes in Computer Science*, pages 577–593. Springer.

[20] Li, J., Monroe, W., and Jurafsky, D. (2016). Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

[21] Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. (2017). Program induction by rationale generation: Learning to solve and explain algebraic word problems. *CoRR*, abs/1705.04146.

[22] Liu, Y., Sun, C., Lin, L., and Wang, X. (2016). Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

[23] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

[24] Minervini, P. and Riedel, S. (2018). Adversarially regularising neural NLI models to integrate logical background knowledge. In *CoNLL*, pages 65–74. Association for Computational Linguistics.

[25] Munkhdalai, T. and Yu, H. (2016). Neural semantic encoders. *CoRR*, abs/1607.04315.

[26] Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., and Rohrbach, M. (2018). Multimodal explanations: Justifying decisions and pointing to the evidence. *CoRR*, abs/1802.08129.

[27] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM.

[28] Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kociský, T., and Blunsom, P. (2015). Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

[29] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In *ICLR (Poster)*.

[30] Wang, W., Tang, B., Wang, R., Wang, L., and Ye, A. (2019). A survey on adversarial attacks and defenses in text. *CoRR*, abs/1902.07285.

[31] Zhang, W. E., Sheng, Q. Z., Alhazmi, A., and Li, C. (2019a). Adversarial attacks on deep learning models in natural language processing: A survey.

[32] Zhang, W. E., Sheng, Q. Z., and Alhazmi, A. A. F. (2019b). Generating textual adversarial examples for deep learning models: A survey. *CoRR*, abs/1901.06796.

[33] Zhao, Z., Dua, D., and Singh, S. (2018). Generating natural adversarial examples. In *ICLR (Poster)*. OpenReview.net.

A Supplemental Material

A.1 Entailment Templates

List of manually created templates for generating inconsistent explanations. "token1/token2" means that a separate sentence has been generated for each of the tokens. [X] and [Y] are the key elements that we want to identify and use in the other templates in order to create inconsistencies. [...] is a placeholder for any string, and its value is not relevant.

- [X] is/are a type of [Y]
- [X] implies [Y]
- [X] is/are the same as [Y]
- [X] is a rephrasing of [Y]
- [X] is a another form of [Y]
- [X] is synonymous with [Y]
- [X] and [Y] are synonyms/synonymous
- [X] can be [Y]
- [X] and [Y] is/are the same thing
- [X] then [Y]
- [X] if [X] , then [Y]
- [X] so [Y]
- [X] must be [Y]
- [X] has to be [Y]
- [X] is/are [Y]

Neutral Templates

- not all [X] are/have [Y]
- not every [X] is/has [Y]
- just because [X] does not/n't mean/imply [Y]
- [X] is/are not necessarily [Y]
- [X] does not/n't have to be [Y]
- [X] does not/n't imply/mean [Y]

Contradiction Templates

- [...] cannot/can not/can n't [X] and [Y] at the same time/simultaneously
- [...] cannot/can not/can n't [X] and at the same time [Y]
- [X] is/are not (the) same as [Y]
- [...] is/are either [X] or [Y]
- [X] is/are not [Y]
- [X] is/are the opposite of [Y]
- [...] cannot/can not/can n't [X] if [Y]
- [X] is/are different than [Y]
- [X] and [Y] are different [...]