HILDIF: Interactive Debugging of NLI Models Using Influence Functions

Hugo Zylberajch, Piyawat Lertvittayakumjorn, Francesca Toni
Department of Computing, Imperial College London, UK
{hz1820, pl1515, ft}@imperial.ac.uk

Abstract

Biases and artifacts in training data can cause unwelcome behavior in text classifiers (such as shallow pattern matching), leading to lack of generalizability. One solution to this problem is to include users in the loop and leverage their feedback to improve models. We propose a novel explanatory debugging pipeline called **HILDIF**, enabling humans to improve deep text classifiers using influence functions as an explanation method. We experiment on the Natural Language Inference (NLI) task, showing that HILDIF can effectively alleviate artifact problems in fine-tuned BERT models and result in increased model generalizability.

1 Introduction

Given two sentences, a premise and a hypothesis, Natural Language Inference (NLI) is the task of determining whether the premise entails the hypothesis, and it has been considered by many as a sign of language understanding (Condoravdi et al., 2003; Dagan et al., 2005). Although recent deep learning models have shown to achieve good performances on different NLI datasets, as in other tasks, they have been shown to learn shallow heuristics. For example, a model is very likely to predict *entailment* for all hypotheses constructed from words in the premise (McCoy et al., 2019). A key challenge is therefore to understand when and why state-of-the-art NLI models fail and try to mitigate the problems accordingly.

In order to bring to light this kind of pathology, one can use explanation techniques to comprehend how a black box model makes particular predictions. For instance, feature attribution methods explain by identifying parts of inputs that mainly contribute to predictions (Smilkov et al., 2017; Sundararajan et al., 2016; Ribeiro et al., 2016; Lundberg and Lee, 2017). Further, example-based methods, such as influence functions (Koh and Liang, 2017), identify training data points which are the most important for particular predictions. Existing works have proposed ways to improve models by incorporating human feedback, in response to the explanations, by: adding model constraints by fixing certain parameters (Stumpf et al., 2009; Lertvittayakumjorn et al., 2020), adding training samples (Teso and Kersting, 2019), and adjusting models' weights directly (Kulesza et al., 2015).

In this paper, we propose a novel interactive model debugging pipeline called **HILDIF** – **H**uman **I**n the **L**oop **D**ebugging using **I**nfluence **F**unctions. With the NLI task as a target, we use influence functions as an explanation method to help users understand the model reasoning via influential training examples. Then, for each influential example shown, the users provide feedback to create augmented training samples for fine tuning the model. Using HILDIF, we effectively mitigate artifact issues of BERT models (Devlin et al., 2019) trained on the MNLI dataset (Williams et al., 2018) and tested on the HANS dataset (McCoy et al., 2019), which is a known pathological setting for most deep NLI models working on English language. Our code can be found at https://github.com/hugozylberajch/HILDIF.

2 Related Work

Influence Functions. Introduced by Hampel (1974), influence functions compute how upweighting individual examples in the training loss changes the model parameters. Influential training examples can also be used to study models (Koh and Liang, 2017). They are particularly useful when feature attribution scores are not sufficient to illustrate how the model *reasons*. In the NLI task, for example, single input words may not suffice to explain a certain prediction, and the overall semantics and structures in the input may be needed.

Proceedings of the First Workshop on Interactive Learning for Natural Language Processing , pages 1–6

Recently, Han et al. (2020) showed that influence functions can capture key fine-grained interactions among input words and detect the presence of artifacts that lead to incorrect NLI predictions.

Although very appealing, influence functions are computationally expensive. Hence, Koh and Liang (2017) reduced computational complexity by using the **LI**near time **S**tochastic **S**econd order **A**lgorithm (LISSA) for calculating approximations. Guo et al. (2020) proposed FASTIF, which further speeds up the calculation using the k-nearest neighbors algorithm. They also fine tuned the model with influential training samples of *anchor points* (i.e., some data points in the validation set) to correct model errors. We will use FASTIF as a tool to explain BERT model's predictions on the NLI task in our experiment.

Explanatory Interactive Debugging, where we improve a model by leveraging user feedback after presenting explanations for model predictions, was first introduced using simple statistical models such as Naïve Bayes models or Support Vector Machines with simple explanatory techniques (Stumpf et al., 2009). Recently, explanatory debugging has been applied to more complex models using refined interpretability methods. In FIND (Lertvittayakumjorn et al., 2020), a masking matrix is added at the end of a CNN text classifier so as to disable particular CNN filters based on human feedback in response to LRP-based explanations (Arras et al., 2016). In CAIPI (Teso and Kersting, 2019), the user investigates and corrects a LIME-based explanation (Ribeiro et al., 2016) for each prediction. Then additional training samples, created based on the correction, are used to fine tune the model. For more details on explanatory debugging, we refer interested readers to the survey by Lertvittayakumjorn and Toni (2021).

As in CAIPI, we will exploit user feedback to control the generation of augmented samples for fine tuning the model. However, our explanations are influential training samples which are more suitable for explaining NLI predictions. This is an improvement from Guo et al. (2020) that simply fine tuned the model on influential samples without human feedback involved.

3 HILDIF

We propose in Algorithm 1 a new pipeline called **HILDIF** (**H**uman **I**n the **L**oop **D**ebugging with **I**nfluence **F**unctions) for debugging deep text classifiers using influence functions. As far as our knowledge goes, this is the first interactive explanatory debugging algorithm that makes effective use of influence functions. To improve a model f using HILDIF, a set of anchor points $\mathcal{X} = (x_1, x_2, ..., x_n)$ is first selected from the validation dataset $\mathcal{V}$, and the predictions $\hat{\mathcal{Y}} = (\hat{y_1}, \hat{y_2}, ..., \hat{y_n})$ are computed using the model f. Then, for each anchor point x_i, we use FASTIF to identify p influential training samples $\mathcal{Z}_i = (z_{i1}, z_{i2}, ..., z_{ip})$, and we define $\mathcal{Z}$ as a collection of $\mathcal{Z}_i$ for all $x_i \in \mathcal{X}$. Next, for each pair of $(x_i, z_{ij}), i \in \{1, ..., n\}, j \in \{1, ..., p\}$, the user will give a score of similarity s_{ij} that will be used to generate synthetic data using a data augmentation function g. Finally, the model is fine tuned on the new generated data samples.

Algorithm 1: HILDIF. $\mathcal{L}$ is a labeled training set, $\mathcal{V}$ is a labeled validation set, T is the number of iteration, and g is a data augmentation method.

```
t ← 0
f ← FIT(L)
while t < T do
    X ← SELECT_ANCHORS(f, V)
    Ŷ ← f(X)
    Z ← EXPLAIN(f, X, Ŷ)
    S ← ∅
    for x_i ∈ X do
        for z_ij ∈ Z_i do
            Present x_i, ŷ_i, z_ij to the user;
            Obtain a similarity score s_ij for
             the influential example z_ij;
            S ← S ∪ g(z_ij, s_ij)
    f ← FINE_TUNE(f, S)
    t ← t + 1
Return : f
```

Next, we explain, in detail, each step of HILDIF, including explanation generation, user feedback collection, and data augmentation.

Explanation Generation. From the validation set $\mathcal{V}$, we can either select anchor points randomly or handpick some that contain particular heuristics we want to debug. After that, the user is presented with a list of top-p most negatively influential training data points for each anchor point. These influential data points contribute to the decrease of the model's loss when upweighted. Hence, fine-tuning the model using these data points should improve the model performance as studied by Guo et al.

(2020). However, since HILDIF relies on FASTIF which only approximates influence scores, we hypothesize that we can achieve better performance by asking humans to assess relevancy of the influential training samples returned by FASTIF before fine-tuning.

User Feedback Collection. For each anchor point x_i and corresponding influential sample z_{ij}, the user is asked the question: *The test case and the presented sample are: (1) Very different; (2) Different; (3) Can't decide; (4) Similar; (5) Very similar*; the user can then answer by selecting a radio button. Then z_{ij} will obtain a similarity score s_{ij} from 1 to 5 accordingly based on the user's answer. *Similar* in this context means that both samples share the same type of heuristics or lexical artifacts.

Data Augmentation. To create an augmented sample for the NLI task, we have to make sure that the overall semantics of the premise and the hypothesis as well as the overall relation between the two sentences are preserved. We therefore choose random word replacement with synonyms as well as back translation for data augmentation since neither changes the semantic of the sentences. Moreover, we found empirically by testing different configurations that generating $10 \times s_{ij}$ augmented samples for the influential sample z_{ij} yielded the best results. For instance, an influential sample with the score 3 leads to 30 augmented samples with the same label as the original sample.

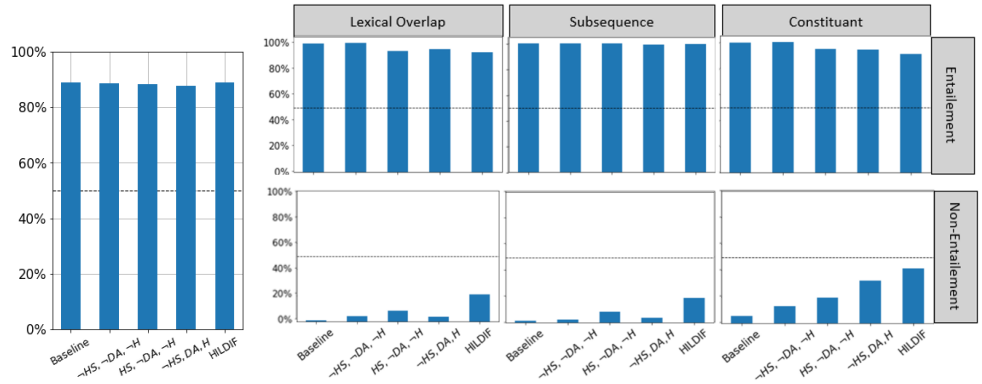
4 Experimental Setup

Datasets and Models. We evaluate our pipeline with a pretrained BERT-base cased model. We use the MNLI dataset (Williams et al., 2018) for training and validation, and the HANS dataset, which is known to be a dataset where BERT performs poorly (McCoy et al., 2019), for testing. For the MNLI training and validation set, we merge the class *neutral* and *contradiction* into a single *non-entailment* class, following the HANS dataset's setting. HANS targets three heuristics of NLI and includes examples showcasing these heuristics: *Lexical Overlap* where the hypothesis is constructed with words from the premise, *Constituent*, where the hypothesis is a subtree of the premise's parse tree, and *Subsequence*, where the hypothesis is a contiguous subsequence of the premise (see Table 1 in the Appendix for some examples). NLI models almost always predict entailment for any example containing these heuristics although sometimes the correct label is non-entailment. So, our goal is to make the model better detect non-entailment cases while maintaining its performance on the entailment cases. For the overall performance, we chose accuracy as our evaluation metric because HANS is a balanced dataset (containing, for each subgroup of heuristics, 5,000 samples of the entailment class and 5,000 samples of the non-entailment class).

Implementation Details. All our models are implemented using the pytorch library and trained using the AdamW optimizer. The HANS dataset is held-out during training and fine-tuning and is only used for testing. For computing influence functions, we use the FASTIF algorithm and FAISS library (Johnson et al., 2019) for k-nearest neighbors search. Finally, we ran all our experiment on a single 12GB NVIDIA Tesla K80 GPU. With this setting, the computation of influence scores of 5,000 training points for a corresponding anchor point takes approximately seven minutes. The BERT-base model is trained for two epochs on the MNLI training dataset.

Regarding user feedback collection, due to human resources constraints, we did our interactive experiments with one expert user. Further experiments could be conducted with more users, and results for the same pair of anchor point and influential point could be aggregated in order to reduce human bias.

Comparison. We experimented with $T = 1$, using five anchor points with 10 and 20 influential samples each. We introduce three binary propositions that will define the debugging pipeline: HS: Human scoring, DA: Data augmentation, and H: Handpicked anchor points. Without human scoring ($\neg HS$), every influential sample receives a score of 5. Without data augmentation ($\neg DA$), the fine tuning is done on each influential sample only, and without handpicked anchor points ($\neg H$), anchor points are selected randomly. Note that our handpicked anchor points were chosen among the validation samples that contain either the lexical overlap or the subsequence heuristic (see Table 2 in the Appendix). We compared the performance of eight different configurations of debugging algorithms that stem from these three binary propositions. For each configuration, we trained and im-



(a) (b)

Configuration			10 influential points			20 influential points		
			LO	SUB	CON	LO	SUB	CON
HS	DA	H	(93.99 , 18.16)	(98.90 , 12.10)	(92.99 , 36.10)	(92.70 , **20.76**)	(99.24 , **18.80**)	(90.82 , **41.58**)
		$\neg H$	(98.89 , 3.56)	(99.66 , 2.74)	(97.12 , 12.10)	(98.12 , 2.06)	(98.51 , 4.20)	(97.22 , 19.21)
	$\neg DA$	H	(95.51 , 6.81)	(96.40 , 5.32)	(93.71 , 22.28)	(98.94 , 8.64)	(97.81 , 3.40)	(95.01 , 19.32)
		$\neg H$	(98.81 , 2.16)	(98.73 , 2.25)	(95.55 , 16.12)	(99.09 , 1.91)	(98.89 , 2.12)	(93.39 , 17.64)
$\neg HS$	DA	H	(99.42 , 2.94)	(99.65 , 3.15)	(96.01 , 32.15)	(99.32 , 3.76)	(99.20 , 4.01)	(94.99 , 32.34)
		$\neg H$	(99.10 , 1.86)	(99.81 , 2.84)	(97.67 , 9.66)	(98.10 , 2.40)	(98.89 , 3.90)	(96.20 , 17.01)
	$\neg DA$	H	(98.13 , 3.10)	(99.62 , 2.87)	(97.03 , 9.99)	(97.21 , 4.01)	(99.68 , 2.61)	(96.54 , 13.13)
		$\neg H$	(99.49 , 0.90)	(99.32 , 1.67)	(97.88 , 6.10)	(99.12 , 1.12)	(99.01 , 1.29)	(97.15 , 5.99)
Baseline			**LO**: (99.56 , 0.98)		**SUB**: (100.00 , 1.30)		**CON**: (99.02 , 5.74)	

(c)

Figure 1: HS stands for Human Scoring, DA for Data Augmentation and H for Handpicked anchor points. (a) Average accuracy on the MNLI test set (b) Accuracies on the HANS evaluation set, which has 3 heuristic categories and 2 classes. Dashed lines show chance performance. (c) Accuracies on the HANS evaluation set for different configurations of all the debugging procedures. LO stands for Lexical Overlap, SUB for Subsequence, and CON for Constituent category of heuristics. For each cell, the first value of the tuple is the accuracy on the entailment class and the second is the accuracy on the non-entailment class. Best scores for the non-entailment class in bold.

proved three models using different random seeds and averaged the final performance on the test set. Note that the $(\neg HS, \neg DA, \neg H)$ configuration is the algorithm used in Guo et al. (2020) whereas the (HS, DA, H) and $(HS, DA, \neg H)$ configurations are our HILDIF algorithm.

5 Results

Figure 1b shows the accuracies on the HANS dataset of the baseline model (i.e., BERT trained on MNLI), and of four configurations, including (HS, DA, H) which is displayed as HILDIF in the figure. We can see that HILDIF consistently achieved a higher accuracy in all three categories of heuristics for the non-entailment class and a slightly lower accuracy for the entailment class. Actually, we observe the trade-off between the accuracies of both classes in all the four configurations. However, HILDIF still got higher overall accuracy on the HANS dataset than the baseline and the other configurations. Moreover, interactive debugging with human scores yielded better accuracies than debugging without human scores for the Lexical Overlap and the Subsequence categories. Meanwhile, on the Constituent category, handpicking anchor points with targeted heuristic led to a big jump in accuracy that outperformed the configuration with human feedback but random anchor points. Therefore, incorporating human knowledge since the selecting anchors step is also helpful when we have prior knowledge about the model bugs. Note also, in Figure 1a, that the model accuracies on the MNLI for HILDIF and the other configurations stay close to the baseline model's accuracy, as desired.

The table in Figure 1c shows that fine tuning the model with augmented data samples, instead of the influential samples only, gave better results in most cases. This was likely because data augmentation could help prevent the model from overfitting the influential samples. Besides, there was little to no

improvement in the model performance when we added user feedback (i.e., human scores) for random anchor points but a substantial improvement for handpicked anchor points. This can be because, during user feedback collection, most of the data samples are difficult to compare as they either satisfy several heuristics or no heuristics relevant to the NLI task. When looking at some influential samples for handpicked anchor points, most satisfy the same heuristic and, if not, they can be easily spotted by human eyes. Although we are still far from chance performance on the non-entailment class, HILDIF achieved a substantial increase in accuracy with just five anchor points.

6 Conclusion

We introduced HILDIF, an interactive explanatory debugging pipeline for deep text classifiers, and ran experiments on the NLI task, achieving high accuracies with MNLI-trained BERT across all categories of the pathological HANS dataset. Future work includes enhancement of the data augmentation part, including the use of a variational autoencoder or a GPT-2 based generative model for synthetic data generation. Also, with more human resources, experiments can be conducted by fine-tuning more than one iterations ($T > 1$) with more anchor points for each iteration. Finally, it would be interesting to apply HILDIF to other text classification tasks, given that, except for the handpicked anchor points that are chosen with knowledge of the task, every step of the pipeline is task-independent.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany. Association for Computational Linguistics.

Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 38–45.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2020. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*.

Frank R. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online. Association for Computational Linguistics.

J. Johnson, M. Douze, and H. Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, pages 1–1.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, page 126–137, New York, NY, USA. Association for Computing Machinery.

Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–348, Online. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn and Francesca Toni. 2021. Explanation-based human debugging of nlp models: A survey. *arXiv preprint arXiv:2104.15135*.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International journal of human-computer studies*, 67(8):639–662.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2016. Gradients of counterfactuals. *arXiv preprint arXiv:1611.02639*.

Stefano Teso and Kristian Kersting. 2019. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 239–245, New York, NY, USA. Association for Computing Machinery.

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, New Orleans, Louisiana. Association for Computational Linguistics.

A Examples and Anchor Points

Table 1 shows an example from the MNLI dataset (Williams et al., 2018) and three examples from the HANS dataset (each of which has a different heuristic type) (McCoy et al., 2019). Besides, Table 2 shows the five handpicked anchor points used in the experiment.

MNLI Dataset
$\mathcal{P}$: News ' cover says the proliferation of small computer devices and the ascendance of Web-based applications are eroding Microsoft's dominance.
$\mathcal{H}$: Microsoft is a more profitable company than Apple.
Label: non-entailment
HANS Dataset (Lexical Overlap)
$\mathcal{P}$: The professors advised the judge.
$\mathcal{H}$: The judge advised the professors.
Label: non-entailment
HANS Dataset (Subsequence)
$\mathcal{P}$: The professor who introduced the doctors recognized the secretaries.
$\mathcal{H}$: The doctors recognized the secretaries.
Label: non-entailment
HANS Dataset (Constituent)
$\mathcal{P}$: Certainly the senators recognized the actor.
$\mathcal{H}$: The senators recognized the actor.
Label: entailment

Table 1: Examples of premises and hypotheses from the MNLI dataset and the HANS dataset. $\mathcal{P}$ and $\mathcal{H}$ stand for premise and hypothesis, respectively.

$\mathcal{P}$: Similar conclusions have been reached by state legal needs ' studies in a dozen states including Florida , Georgia , Hawaii , Illinois , Indiana , Kentucky , Maryland , Massachusetts , Missouri , Nevada , New York , and Virginia , using a variety of methodologies for estimating the unmet legal needs of the poor.
$\mathcal{H}$: Similar conclusions have been reached by state legal needs ' studies.
Label: entailment
Heuristics: Subsequence
$\mathcal{P}$: From Cockpit Country to St . Ann ' s Bay.
$\mathcal{H}$: From St . Ann ' s Bay to Cockpit Country.
Label: non-entailment
Heuristics: Lexical Overlap, Reverse Ordering
$\mathcal{P}$: Shoot only the ones that face us , Jon had told Adrin.
$\mathcal{H}$: Shoot the ones that face us , Adrin told Jon.
Label: non-entailment
Heuristics: Lexical Overlap, Reverse Ordering
$\mathcal{P}$: Just north of the Shalom Tower is the Yemenite Quarter , its main attractions being the bustling Carmel market and good Oriental restaurants.
$\mathcal{H}$: The Shalom Tower is north of the Yemenite Quarter.
Label: non-entailment
Heuristics: Lexical Overlap, Reverse Ordering
$\mathcal{P}$: Life , unlike Reich ' s book , is not a series of morality fables.
$\mathcal{H}$: Reich ' s book is a series of morality fables.
Label: entailment
Heuristics: Lexical Overlap, Negation

Table 2: Five handpicked anchor points from the MNLI validation set, with specific heuristics. $\mathcal{P}$ and $\mathcal{H}$ stand for premise and hypothesis, respectively.