When Can Models Learn From Explanations?
A Formal Framework for Understanding the Roles of Explanation Data

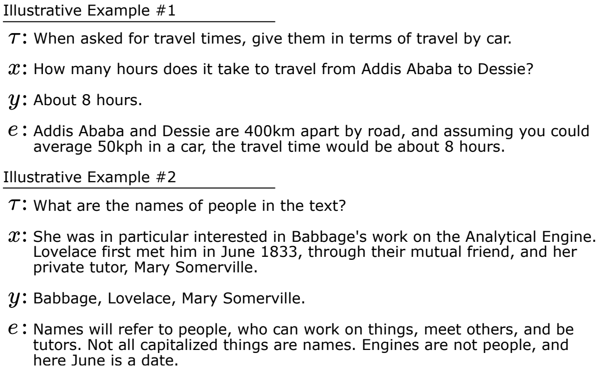
Peter Hase and **Mohit Bansal**
Department of Computer Science
University of North Carolina at Chapel Hill
{peter, mbansal}@cs.unc.edu

Abstract

Many methods now exist for conditioning models on task instructions and user-provided explanations for individual data points. These methods show great promise for improving task performance of language models beyond what can be achieved by learning from individual (x, y) pairs. In this paper, we (1) provide a formal framework for characterizing approaches to learning from explanation data, and (2) we propose a synthetic task for studying how models learn from explanation data. In the first direction, we give graphical models for the available modeling approaches, in which explanation data can be used as model *inputs*, as *targets*, or as a *prior*. In the second direction, we introduce a carefully designed synthetic task with several properties making it useful for studying a model's ability to learn from explanation data. Each data point in this binary classification task is accompanied by a string that is essentially an answer to the *why* question: "why does data point x have label y?" (Miller, 2019). We aim to encourage research into this area by identifying key considerations for the modeling problem and providing an empirical test bed for theories of how models can best learn from explanation data.[1]

1 Introduction

A long line of past work has sought to use free-text explanations, rationales, and other similar data to improve machine learning models. Proposed methods use explanations to constrain or regularize the learned model (Zaidan et al., 2007; Small et al., 2011; Ba et al., 2015; Zhang et al., 2016; Srivastava et al., 2017; Liang et al., 2020), to automatically label data for data augmentation (Hancock et al., 2018; Wang et al., 2019a; Awasthi et al., 2020), as additional supervision (Narang et al., 2020; Hase et al., 2020; Pruthi et al., 2021) or intermediate structured variables (Camburu et al., 2018; Rajani et al., 2019; Wiegreffe et al., 2020), and simply as model inputs (Rupprecht et al., 2018; Co-Reyes et al., 2019; Zhou et al., 2020).



Illustrative Example #1

$\mathcal{T}$: When asked for travel times, give them in terms of travel by car.

x: How many hours does it take to travel from Addis Ababa to Dessie?

y: About 8 hours.

e: Addis Ababa and Dessie are 400km apart by road, and assuming you could average 50kph in a car, the travel time would be about 8 hours.

Illustrative Example #2

$\mathcal{T}$: What are the names of people in the text?

x: She was in particular interested in Babbage's work on the Analytical Engine. Lovelace first met him in June 1833, through their mutual friend, and her private tutor, Mary Somerville.

y: Babbage, Lovelace, Mary Somerville.

e: Names will refer to people, who can work on things, meet others, and be tutors. Not all capitalized things are names. Engines are not people, and here June is a date.

Figure 1: Hypothetical data and explanations. Here, x is an input that one might expect a model to produce the correct output for after fitting to (x, y) pairs. For some models, x may be sufficient, while others may benefit from additional information provided by e.

However, there are many tasks in NLP where improvements in performance prove elusive even when using thousands of explanations as additional data (Narang et al., 2020; Hase et al., 2020). A few observations could explain this situation: (1) the modeling space has not been fully explored for these tasks, but improvements are possible; (2) pretrained language models already store the knowledge that the explanations would have provided, so they do not need them; (3) the language models do not need any information that is not already learnable from the task's input-output pairs. We do not yet know which explanation is best, and therefore it would be helpful to more deeply understand the motivations behind existing modeling approaches.

In this paper, we (1) present a formal framework for characterizing approaches to learning from explanation data, and (2) we propose a synthetic task for studying how models learn from natural language data. Specifically, we first present graphical

[1] Our code and data are publicly available at: https://github.com/peterbhase/ExplanationRoles. An extended technical report on this topic is available at: https://arxiv.org/abs/2102.02201.

Proceedings of the First Workshop on Learning with Natural Language Supervision, pages 29 - 39
May 26, 2022 ©2022 Association for Computational Linguistics

models for various approaches where explanation data is used either as model *inputs*, *targets*, or *priors*, and we characterize existing methods according to these graphical models. Then, based on past results, we suggest which models might be most appropriate for explanation data. Next, we present a synthetic task which shares important properties with NLP tasks involving explanation data. Constructing this task helps us carefully specify the manner in which we expect explanations to be useful to models. We provide simple experimental verification that the task is solvable by existing Transformer models when using explanations as additional data but very difficult to solve without them. Our aim is to outline promising approaches in the area and contribute a concrete test bed to assist others in developing new models for learning from natural language explanations.

2 Formalizing the Roles of Explanations

In what follows, we discuss our framework for modeling with explanations and relevant work (Sec. 2.1), as well as promising approaches for learning from explanations (Sec. 2.2).

What Is an Explanation? We use the term "explanation" to refer to the data one might collect if asking a person to answer the question, "Why does data point x have label y?" This is a formulation of the explanation as an answer to a *why-question* of the kind discussed in Miller (2019). Rather than try to give a formal definition of the kind of data generated from this question, we proceed with some illustrative examples, shown in Fig. 1.

2.1 Formal Framework and Relevant Work

In this section, we lay out our theory of how explanations may be used in modeling a task, in a standard supervised learning setup for obtaining a MAP estimate of model parameters:

$$\hat{\theta} = \arg\max_{\theta} p(\theta|X, Y)$$
$$p(\theta|X, Y) \propto p(Y|X, \theta)p(\theta)$$

where Y is a set of labels for inputs X. We refer to the role of Y in this probabilistic model as the *target*, X as an *input*, and $p(\theta)$ as a *prior*. Below we describe existing approaches to adding explanations into this framework. An overview of the corresponding graphical models is shown in Fig. 2.

Using Explanations as Targets. Explanations are often used as additional supervision (shown as Multi-Task in Fig. 2). For instance, Pruthi et al. (2021) consider using attention weight explanations (from a model) as targets in a multi-task framework, and they observe accuracy improvements in what is essentially model distillation. Meanwhile, natural language explanations appear as targets in a multi-task framework, using datasets with explanations for each data point (Camburu et al., 2018; Narang et al., 2020; Hase et al., 2020; Wiegreffe et al., 2020). None of these works find improvements in task performance from incorporating explanations. It is perhaps even concerning that a model could learn to generate coherent "explanations" without the learning of this ability influencing the models that are found for the task.

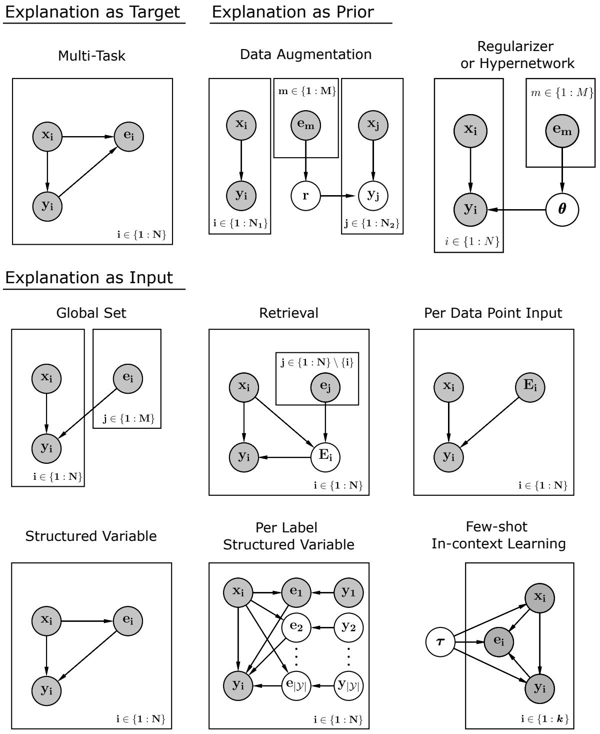


Figure 2: Graphical models for several approaches to using explanations as *targets*, as *inputs*, and as *priors*. Typically past works do not condition on human-given explanations at test time, unless they are designed to not leak the data point label.

Using Explanations as Inputs. Additional inputs may be valuable for solving some tasks. One family of approaches uses explanations as model inputs for each data point (Per Data Point Input in Fig. 2). Talmor et al. (2020) systematically study RoBERTa's ability to combine pieces of knowledge for a task by including relevant factoids in the text input. Co-Reyes et al. (2019) provide online natural language feedback to RL agents, and Rupprecht et al. (2018) take a similar approach to interactive image segmentation with language feedback.

More commonly, approaches do not use human explanations at test time. In ExpBERT (Murty et al., 2020), a model conditions on vector representations of an input x and a single "global" set of explanations in order to make each prediction (Global Set in Fig. 2). This approach may not scale well to large numbers of explanations, however. Zhou et al. (2020) treat explanations as latent variables, and at inference time they retrieve explanations from the training data (Retrieval in Fig. 2). A number of works condition on explanations generated at test time using generative models learned with human explanations as supervision, which are represented as Structured Variable and Per-Label Structured Variable in Fig. 2 (Camburu et al., 2018; Rajani et al., 2019; Kumar and Talukdar, 2020; Hase et al., 2020; Wiegreffe et al., 2020; Zhao and Vydiswaran, 2021). While such structured variables could be useful in principle, these methods have not produced sustained improvements in model accuracy.

Lastly, large language models have recently opened the door for using explanations in few-shot in-context learning (Brown et al., 2020). We represent this approach as Few-shot In-context Learning in Fig. 2. We do not draw the dependencies between distinct data points in the context that would be implied by the attention graph of Transformers, but instead represent the dependence of each data point on the unknown task τ, which models evidently do inference over at test time. Initial work in this direction suggests that models of a sufficiently large size (280B parameters) can learn from explanations provided in a few-shot in-context learning setting (Lampinen et al., 2022).

Using Explanations as Priors. We group together approaches to defining a distribution over model parameters, including those conditioning on data, $p(\theta|data)$. This is a prior over model weights not in the sense that the distribution is independent of data (which it is not), but rather that the posterior parameters are conditioned on the prior. Explanations have been used to constrain the learned model (Srivastava et al., 2017, 2018) or to place priors over how features are weighted or extracted (Zaidan et al., 2007; Small et al., 2011; Zhang et al., 2016; Ross et al., 2017; Bao et al., 2018; Selvaraju et al., 2019; Liang et al., 2020; Stammer et al., 2020; Pruthi et al., 2021; Stacey et al., 2022). Other works map directly from text to model parameters (Ba et al., 2015; Andreas et al., 2018). These methods are all effectively described by Regularizer or Hypernetwork in Fig. 2. Lastly, a few approaches learn to use explanations for automatically labeling data for data augmentation purposes (Hancock et al., 2018; Wang et al., 2019b; Awasthi et al., 2020), which is effectively fitting to data from a prior distribution given by the labeling mechanism (Data Augmentation in Fig. 2).

2.2 Promising Models

Based on our review of existing approaches, we make a few key observations that we believe will assist in the design of future techniques:

1. Using free-text explanations as structured variables and as targets do not appear to be promising approaches at the moment (Hase et al., 2020; Narang et al., 2020).

2. Free-text explanations may be useful as priors in computer vision (Liang et al., 2020), but we know of no successful use case for tasks besides Stacey et al. (2022), which effectively reduces free-text explanations to a bag of words.

3. The only cases we know of where free-text explanations improve model performance on NLP tasks is when they are used as model inputs via the Global Set model, (Murty et al., 2020) a Retrieval model (Zhou et al., 2020), and an In-Context Learning model using 280B parameters (Lampinen et al., 2022).

The upshot of these results is that the most promising approaches for learning from explanation data are likely those treating explanations as inputs (in a manner that does not require new explanations at test time). However, we recommend that other graphical models not be ruled out completely, in case there are promising methods in those families that have yet to be explored.

3 Synthetic Task

Following recent work using synthetic data to investigate sequence modeling questions (Liu et al., 2021; Lovering et al., 2021), we design a synthetic dataset so that we can carefully control several important data properties. In Fig. 3, we show an example data point and description of how it gets its label. The premise of our task is to classify sequences by counting different integers in them.

Core Idea Behind Data. We wish to design a task where, for a data point (x, y), an explanation

Synthetic Task

$\mathcal{T}$: Count whether there are more of integer a than integer b

x: 962 1 80 80 34 40 99 67 50 27 27 17 17 17 17 17 17 53 17 54

y: 1

e: (962, 80, 40, 17, 27)

Description: The sequence x has label 1 because there are more 80s than 40s. The index 962 maps to (80, 40, 17, 27), and indicator 1 says to count (80, 40) rather than (17, 27). If there were more 40s than 80s, the label would be 0. There is a one-to-one map between index values and $e = (index, m, n, r, d)$ tuples.

Analogous Components to Real Data

Index 962 ↔ An easily computable feature connecting the input to its explanation

Indicator 1 ↔ A feature indicating what information from the explanation is relevant for the input's label

e: (962, 80, 40, 17, 27) ↔ An explanation that says why the input received its label, when understood properly

Figure 3: An example of our synthetic task.

e communicates information about *why* input x receives label y. The premise of the task is that a binary label for a sequence of integers x is determined by whether there are more of an integer a in the sequence than there are of an integer b. We refer to integers (a, b) that need to be counted as the *label reason*. This *label reason* forms the basis of the explanation for each data point, and it is always exactly specified by the first two integers in x, which we term the *index* and *indicator*. For every data point x, there is an *explanation* $e = (index, m, n, r, d)$ where the *label reason* is given by either (m, n) or (r, d). Whether the *label reason* is the (m, n) integer pair or the (r, d) pair is dictated by the *indicator*. As represented in Fig. 3, $(a, b) = (m, n)$ if the *indicator* is 1 and $(a, b) = (r, d)$ if the *indicator* is 2. We call the data e an explanation because it is a direct encoding of a natural language explanation for the data (x, y). For the data point in Fig. 3, this natural language explanation is "input x receives label 1 because it contains more 80's than 40's, and we do not need to count 17's or 27's for this sequence."

Proposed Dataset. We describe the proposed dataset using some default data parameters for preliminary experiments, but any specific numbers appearing below are easily adjusted. See Supplement D for the full generative process.

1. Train set: 5000 sequences of 20 integers (including *index* and *indicator*), each accompanied by an explanation. There are 500 unique values of *index* in the dataset drawn from $unif(1, 10000)$, so there are 10 points for each *index*, whose values of $m, n, r,$ and d are drawn from $unif(1, 100)$ while requiring that $m \neq n \neq r \neq d$. The corresponding 10 values of *indicator* are split between 1 and 2. Half of the points have label $y{=}1$, i.e. either $\#m{>}\#n$ or $\#r{>}\#d$, depending on which feature is causal. In each x_i, after $m, n, r,$ and d have been randomly placed into the sequence, unfilled slots are filled with samples from $unif(1, 100)$.
2. Dev set: 10,000 points, none appearing in Train, with the same 500 *index* values, and twice the number of points per *index* as Train.
3. Test set: 50,000 points of similar construction to the Dev set, but with five times the points per *index* as Train.

Analogous Properties to Human-Curated Data. We claim that aspects of our synthetic task are analogous to properties that natural language data might take on, which we represent in Fig. 3. First, e is an explanation in the sense that, when understood properly, it is a plausible answer to the question: "why does point x have label y?" The explanation describes the feature that causes the label, i.e. the integers that should be counted. We suggest that the *index* in a sequence is analogous to the topic of some text or the things it refers to: it is an easily computable feature that connects the input to the appropriate explanation. Meanwhile, the *indicator* is a feature that tells how information from an explanation is relevant to deciding the label. Similarly, an explanation might only be understood in the context of the input it explains.

4 Initial Experiments

We include experiments below that (1) show explanation data is helpful for solving our task and (2) demonstrate why the task is hard without explanation data. We make use of a retrieval-based model similar to Zhou et al. (2020), which learns to retrieve explanations from the training dataset to help with prediction at test time (details in Appendix B and C). This model is composed of a RoBERTa-base classifier (Liu et al., 2019) and a SentenceRoBERTa model used for retrieval (Reimers and Gurevych, 2019). The baseline in our experiments is the RoBERTa classifer on its own.

4.1 Explanation Retrieval Enables a Model to Solve Our Task

Design. Using our default dataset containing one explanation per training point, we measure model accuracy with retrieval in a 3×2 design. There are three conditions for the retrieval model: (1) *fixed*, where the Sentence-RoBERTa retriever is fixed and only the classifier is trained, (2) *learned*, where both classifier and retriever are trained end-

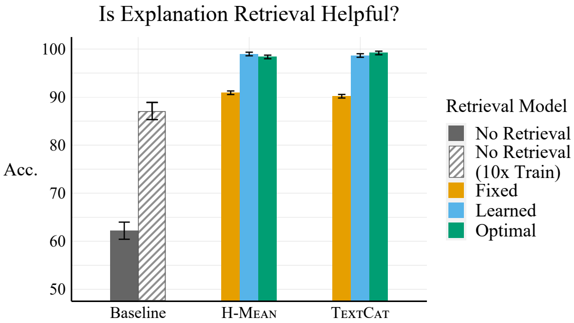


Figure 4: Synthetic task accuracy for our baseline and retrieval model with two conditioning mechanisms, H-MEAN and TEXTCAT.

to-end, and (3) *optimal* where the optimal retrieval model is used and the classifier is trained. We know the optimal retrieval model retrieves explanations with an *index* matching the query point's *index*. The two conditioning mechanisms, H-MEAN and TEXTCAT, differ in how they combine information across multiple retrieved explanations to produce a final prediction (see Appendix B.1).

Results. The results in Fig. 4 show that explanation retrieval can reach accuracies above 98%, improving accuracy by around 37 points over a no-explanation baseline. We also find that the learned retrieval model does as well as the optimal retrieval model, improving over the *fixed* condition by about 7 points. Thus, access to explanations allows the model to perform much better than a no-explanation baseline. In fact, the explanation retrieval model outperforms a no-explanation baseline with as many as 50,000 training data points (a 10x increase), which obtains 87.11% accuracy.

4.2 Why Is The Task Hard Without Explanations?

Design. We measure test accuracy as a function of how many unique explanations (and therefore *label reasons*) there are in the data. While keeping the train set size fixed at 5000 points, we vary how many points share the same explanation $(index, m, n, r, d)$. By default there are 10 points per *index*, and with 5000 points this means that there are 500 unique explanations in the data. We use many as 2500 points per *index*, meaning using two unique explanations. The experiment conditions also vary in how task information is available in the input: (1) for With Explanation, each 20-integer sequence x_i has its explanation appended to it; (2) for No Explanation, only x_i is given, which requires the model to learn the map *index* → (m, n, r, d); (3) for No Index, the *index* is omitted from the input, so the model must infer the *label reason* from the sequence's contents alone.

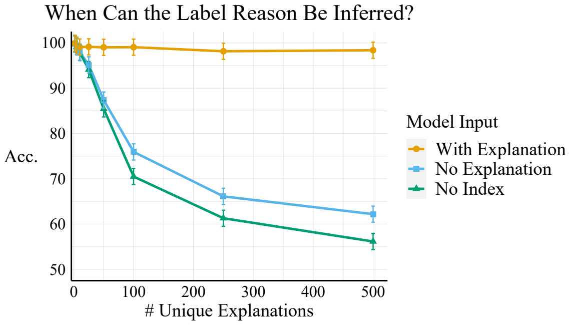


Figure 5: Synthetic task accuracy as a function of the number of unique explanations for data point labels.

Results. The results are shown in Fig. 5. We see that, when the number of unique explanations (and therefore possible *label reasons*) is small, the No Explanation model can achieve an accuracy as high as if it had been directly given the label reason, i.e. as high as the With Explanation condition. Yet, No Explanation model accuracy falls off quickly with the number of unique explanations, reaching accuracies as low as 62.2% with 500 explanations. Evidently, with this many unique explanations, it is too difficult to learn the map between the *index* and the latent *label reason*. Without the *index* in the input (No Index condition), it is even harder to infer the label reason. While accuracy does rise significantly with the size of the training data (see Fig. 4), even using 10x as much train data does not close the gap with the explanation retrieval model.

5 Discussion & Conclusion

We present a synthetic dataset with key similarities to natural language explanation data, and we show that our explanations are highly useful for model learning. However, we emphasize that if a model already "knew" the information in some explanations, it might not need them. This may plausibly occur with sufficiently large pretrained models that store a great deal of factual knowledge (Petroni et al., 2019). Similarly, the necessary information might be learnable from (X, Y) data alone. Future work on modeling approaches we outline in this paper (Fig. 2) will benefit from testing their methods on controlled synthetic tasks as a test of their ability to learn from explanation data. Then, further analysis will be helpful for understanding how explanations contain novel information that is not learned elsewhere in pretraining or finetuning.

Acknowledgements

We thank Miles Turpin and Ethan Perez for helpful discussion of the topics represented here, as well as Xiang Zhou, Prateek Yadav, and our anonymous reviewers for helpful feedback on the work. This work was supported by NSF-CAREER Award 1846185, DARPA Machine-Commonsense (MCS) Grant N66001-19-2-4031, a Google PhD Fellowship, Microsoft Investigator Fellowship, and Google and AWS cloud compute awards. The views contained in this article are those of the authors and not of the funding agency.

Ethical Considerations

There are several positive broader impacts from designing methods for learning from human explanations. Foremost among them is the promise of better aligning learned models with human priors on what kinds of behaviors are good, which could be especially helpful when these priors are hard to robustly encode in supervised learning objectives or unlikely to be learned from the available data. Explanations can also greatly improve model sample efficiency, which is broadly beneficial for difficult, time-consuming, or human-in-the-loop tasks where acquiring a large amount of data is expensive and slow.

There are still some possible risks to this methodology, mainly involving overconfidence in what explanations can provide. For instance, just because explanations improve a model's performance does not mean the model will behave exactly as a human would. We risk anthropomorphizing machine learning models when we suppose their learned interpretations of explanations matches our own.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In *NAACL-HLT 2018*.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. In *ICLR 2020*.

Lei Jimmy Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4247–4255. IEEE Computer Society.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *EMNLP*, pages 1903–1913, Brussels, Belgium. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *NeurIPS 2018*.

John D. Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, Jacob Andreas, John DeNero, Pieter Abbeel, and Sergey Levine. 2019. Guiding policies with language via meta-learning. In *ICLR 2019*.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *ACL*.

Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. 2020. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? In *Findings of EMNLP*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781, Online. Association for Computational Linguistics.

Sawan Kumar and Partha Talukdar. 2020. Nile : Natural language inference with faithful natural language explanations. In *ACL 2020*.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X

Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.

Weixin Liang, James Zou, and Zhou Yu. 2020. ALICE: active learning with contrastive natural language explanations. In *EMNLP*, pages 4380–4391. Association for Computational Linguistics.

Nelson F. Liu, Tony Lee, Robin Jia, and Percy Liang. 2021. Can small and synthetic benchmarks drive modeling innovation? a retrospective study of question answering modeling approaches. *CoRR*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. 2021. Predicting inductive biases of pretrained models.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38.

Shikhar Murty, Pang Wei Koh, and Percy Liang. 2020. Expbert: Representation engineering with natural language explanations. In *ACL*, pages 2106–2113. Association for Computational Linguistics.

Sharan Narang, Colin Raffel, Katherine J. Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. 2020. WT5?! training text-to-text models to explain their predictions. *ArXiv*, abs/2004.14546.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C. Lipton, Graham Neubig, and William W. Cohen. 2021. Evaluating explanations: How much do explanations from the teacher aid students? *TACL*, abs/2012.00893.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *ACL 2019*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *IJCAI*, pages 2662–2670.

Christian Rupprecht, Iro Laina, Nassir Navab, Gregory D. Harger, and Federico Tombari. 2018. Guide me: Interacting with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*.

Ramprasaath Ramasamy Selvaraju, Stefan Lee, Yilin Shen, Hongxia Jin, Shalini Ghosh, Larry P. Heck, Dhruv Batra, and Devi Parikh. 2019. Taking a HINT: leveraging explanations to make vision and language models more grounded. In *ICCV*, pages 2591–2600. IEEE.

Kevin Small, Byron C Wallace, Carla E Brodley, and Thomas A Trikalinos. 2011. The constrained weight space svm: learning with ranked features. In *ICML*, pages 865–872.

Shashank Srivastava, I. Labutov, and T. Mitchell. 2017. Learning classifiers from declarative language. In *NeurIPS 2017*.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *ACL 2018*.

Joe Stacey, Yonatan Belinkov, and Marek Rei. 2022. Supervising model attention with human explanations for robust natural language inference. In *AAAI*.

Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. 2020. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. *CoRR*, abs/2011.12854.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *NeurIPS 2020*.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019a. Does it make sense? and why? a pilot study for sense making and explanation. In *ACL 2019*.

Ziqi Wang, Yujia Qin, Wenxuan Zhou, Jun Yan, Qinyuan Ye, Leonardo Neves, Zhiyuan Liu, and Xiang Ren. 2019b. Learning from explanations with neural execution tree. In *ICLR*.

Sarah Wiegreffe, Ana Marasovic, and Noah A. Smith. 2020. Measuring association between labels and free-text rationales. *CoRR*, abs/2010.12762.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "Annotator Rationales" to Improve Machine Learning for Text Categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267, Rochester, New York. Association for Computational Linguistics.

Ye Zhang, Iain Marshall, and Byron C. Wallace. 2016. Rationale-Augmented Convolutional Neural Networks for Text Classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 795–804, Austin, Texas. Association for Computational Linguistics.

Xinyan Zhao and VG Vydiswaran. 2021. Lirex: Augmenting language inference with relevant explanation. In *AAAI*.

Wangchunshu Zhou, Jinyi Hu, Hanlin Zhang, Xiaodan Liang, Maosong Sun, Chenyan Xiong, and Jian Tang. 2020. Towards interpretable natural language understanding with explanations as latent variables. In *NeurIPS*.

A Additional Experiments

We give additional experimental results with our synthetic dataset in an extended technical report on this topic, available here: `https://arxiv.org/abs/2102.02201`. Additional experiments are conducted to answer a research questions including:

1. Can explanations help models learn to use strong (causal, generalizable) features rather than weak ones?
2. What is the best way to compute explanation representations for prediction?
3. Can models aggregate information across several retrieved explanations?
4. What makes an explanation relevant across data points? What enables a retrieval model to find relevant explanations for a new data point?
5. How does the co-dependence between classifier and retrieval model influence the viability of joint training?
6. Does retrieval of explanations improve model performance on existing natural language datasets?

B Our Model for Initial Experiments

Here, we introduce our chosen model for incorporating explanation data, which makes use of explanations as *model inputs* after they are retrieved from the training data (the "Retrieval" graphical model in Fig. 2). Our approach is similar to Lewis et al. (2020), who marginalize over latent documents retrieved from Wikipedia for question answering, question generation, and fact verification. The marginal distribution is given as:

$$p_\Theta(y|x) = \sum_{e \in \text{top-}k(p_\eta(\cdot|x))} p_\theta(y|x,e)p_\eta(e|x)$$

where top-k gets the top k texts as ranked by the retrieval model, $p_\eta(e|x)$. Note that we never retrieve a data point's own explanation when predicting its label. We do so because explanations can leak the label (Hase et al., 2020) and this approach matches the test-time distribution, where we assume explanations are not collected for new data points (see discussion in Sec. 2).

Zhou et al. (2020) also propose to use explanations as latent variables and retrieve explanations at inference time, but they do not learn the retrieval model, marginalize over the latents during inference, or prohibit data point's own explanations from being retrieved. In our experiments, we compare with their original approach and a version where we marginalize over the latents and learn the retrieval model.

The form of $p_\eta(e|x)$ follows Lewis et al. (2020) and Karpukhin et al. (2020). Given a query x, unnormalized probabilities are computed as:

$$p_\eta(e|x) \propto \exp\left(f_\eta(e)^T f_\eta(x)\right)$$

where f_η embeds each sequence into a vector. To compute top-$k(p_\eta(\cdot|x))$, we search through the training explanations using FAISS (Johnson et al., 2017). We discuss methods for computing $p_\theta(y|x,e)$ and $f_\eta(e|x)$ in Sec. B.1. Because it may be helpful to reason over multiple explanations at once, we extend this model to allow for explanations to be composed into a single "document." Assuming explanations to be conditionally independent given x, we can compute the probability of a set of explanations $E = \{e_c\}_{c=1}^C$ as

$$p(E|x) \propto \exp\left(\sum_{e \in E} f_\eta(e)^T f_\eta(x)\right),$$

where (1) a *context size* C will control the size of the explanation set, (2) a value of k implies that the top Ck will be retrieved, and (3) we sort these Ck explanations into sets in order of their probability $p_\eta(e|x)$.

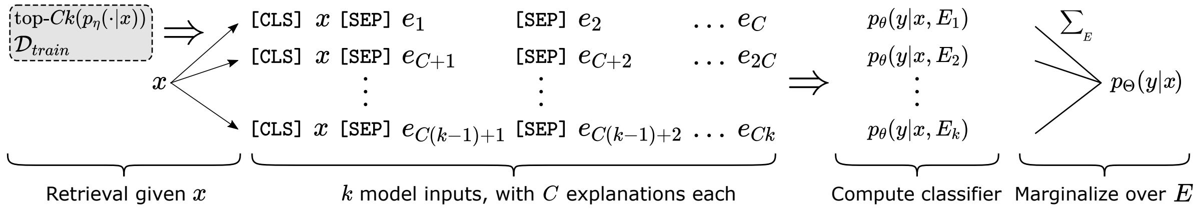

top-$Ck(p_\eta(\cdot|x))$
$\mathcal{D}_{train}$
x
[CLS] x [SEP] e_1 [SEP] e_2 ... e_C
[CLS] x [SEP] e_{C+1} [SEP] e_{C+2} ... e_{2C}
[CLS] x [SEP] $e_{C(k-1)+1}$ [SEP] $e_{C(k-1)+2}$ ... e_{Ck}
$p_\theta(y|x, E_1)$
$p_\theta(y|x, E_2)$
$p_\theta(y|x, E_k)$
$\sum_E$
$p_\Theta(y|x)$
Retrieval given x
k model inputs, with C explanations each
Compute classifier
Marginalize over E

Figure 6: A depiction of our retrieval-based method TEXTCAT. A total of Ck explanations are retrieved and allocated into k latent variables, each a set of explanations E, which are marginalized over to produce a final prediction.

We represent the overall approach in Fig. 6 for one method of computing $p_\theta(y|x, E)$ (described fully in Sec. B.1), where explanations are concatenated with the query sequence. Flowing from left to right, Fig. 6 shows how explanations are retrieved from the training data conditioned on a query sequence x, then allocated into k classifier inputs with C explanations each. The k classifier predictions are aggregated by marginalizing over the latent variable, $Z = E$.

Modeling Assumptions. In using retrieval, we make a few assumptions. First, since the number of forward passes per data point scales with k, we require a relatively small value of k, i.e. $k \leq 10$, for reasonable computational efficiency in SGD-based training. Hence, we must assume that this summation is sufficiently similar to the full summation over latent variables. This assumption is more likely to hold when (1) a small number of documents account for most of the probability mass in $p_\eta(e|x)$, and (2) a pretrained model $p_\eta(e|x)$ yields a decent initial rank-ordering, such that some of the best documents are in the top-k. The exact value of k we use depends on the experiment. A second, more basic assumption is that explanations will be useful in predicting other data points' labels. Such an assumption is needed since we never condition on a data point's own explanation. Lastly, during retrieval we assume that explanations are independent given x, i.e. $p(E|x) = \prod_{e \in E} p(e|x)$. This could be a poor assumption when, for instance, explanations each contribute one of a number of needed facts, in which case it would be helpful to retrieve additional explanations conditioned on what has already been retrieved.

B.1 Conditioning Mechanisms

In this section we describe the methods used to compute $p_\theta(y|x, E)$ and $p_\eta(e|x)$ (see Sec. B for the overall model description). For the classifier $p_\theta(y|x, E)$, we use two methods, TEXTCAT and H-MEAN, which are described below. Then we describe the retrieval model, which is based on Sentence-BERT (Reimers and Gurevych, 2019).

TEXTCAT. Represented in Figure 6, this method takes a straightforward approach to conditioning on a set of explanations: concatenating C explanations and the input x to form a longer sequence of text. Each of the original sequences is separated by a special token, e.g. [SEP] for BERT. In our experiments, we pass this longer sequence into a RoBERTa-base model. After pooling the output token representations, we pass the resulting vector to a 1-layer MLP for classification. We use mean pooling for our synthetic task and NLI; for relation extraction tasks, we concatenate the representations corresponding to the initial tokens in the *subject* and *object* words, since this is an especially effective pooling technique (Baldini Soares et al., 2019).

This approach allows the model to reason over all of the explanations and the input together. While the method may be limited by the fact that some models can face difficulties in processing long pieces of text (Beltagy et al., 2020), this issue is partly mitigated by marginalizing over k sets of explanations. As a result of the marginalization, the final prediction can be conditioned on a far higher number (Ck) of individual explanations than could fit in the context alone.

H-MEAN. By H-MEAN, we refer to the kind of unweighted hidden representation averaging used in Co-Reyes et al. (2019) and Zhou et al. (2020). H-MEAN works by first obtaining representations of the input x and a single explanation e at a time, then passing the unweighted average of these representations to an MLP. For a fair comparison with TEXTCAT, we use the same token pooling and a 1-layer MLP. So with C explanations to condition on, $x' = concatenate(x, e)$, and vector representations from RoBERTa(x'), H-MEAN obtains a single representation as

$$h = \frac{1}{C}\sum_{c=1}^{C} \text{RoBERTa}(x')$$

which is then passed to the MLP for classification. H-MEAN does not face the same sequence length limitations as TEXTCAT, but by separately processing of each explanations H-MEAN may fail to integrate information across explanations. This method also becomes expensive when we marginalize over E (which is what allows retrieval to be learned), as it requires Ck forward passes for a single prediction.

B.2 Retrieval

We use a similar approach to retrieval as in Lewis et al. (2020), namely using vector representations of sequences from a pretrained transformer to compute

$$p_\eta(e|x) \propto \exp\left(f_\eta(e)^T f_\eta(x)\right),$$

which is followed by computing top-$Ck(p_\eta(\cdot|x)$. We use an approximate but sub-linear time search method (FAISS) to find the top-Ck points (Johnson et al., 2017). In our experiments we find that it is necessary to use Sentence-BERT (Reimers and Gurevych, 2019) as our pretrained f_η, rather than simply a pretrained RoBERTa model. Sentence-BERT is a network trained to produce semantic representations of sentences that can be compared under cosine similarity. In our experiments, we use the Sentence-RoBERTa-base model trained on a combination of several NLI and semantic textual similarity tasks, with mean pooling of token representations. We normalize the representations we obtain from this model, so that our inner product is equivalent to a cosine similarity.

Note that during training, we never condition on a data point's own explanation when predicting its label. This is an important constraint for matching the train and test-time distributions. At test time, we assume we have access only to past (training) explanations, since they can be expensive to collect and conditioning on explanations at test time can lead to label leakage, meaning what is essentially the benefit of human labeling could be mistaken as improvements in model performance.

C Training Details

C.1 Runtimes.

Regarding training times, we run most experiments on a single NVIDIA RTX 2080 GPU, with runtimes as follows: 4.0 hours for 40 epochs of the no-retrieval RoBERTa-base using the synthetic dataset; 5.7 hours for 40 epochs of RoBERTa-large in the same setting; 8.6 hours for 20 epochs of learned retrieval with RoBERTa-base models on synthetic data.

C.2 Training Hyperparameters and Analysis

For optimization, we use AdamW with a learning rate of $1\mathrm{e}{-5}$ and gradient norm clipping at norm 1. For the LR, we use a linear warmup and decay schedule peaking at 10% of the training steps for experiments with synthetic data and at 1% for experiments with existing datasets (given the larger training set sizes). The batch size is set to 10 across all experiments.

We decide how often to rebuild the representations of training explanations while learning the retrieval model by tuning across frequency values in the range {10%, 20%, 33%, 50%, 100%} (i.e. to rebuild at this percentage of every epoch), as well as never rebuilding. In our synthetic setting, the only noticeable drop in performance comes from never rebuilding. As long as representations are re-encoded at least as often as every epoch, we notice no difference in final test accuracy, though in early experiments we observed that rebuilding more often improved training stability. To err on the safe side of training stability, we re-encode the representations every 20% of each epoch in all experiments except e-SNLI with full data, where we re-encode every 30% of each epoch.

Additionally, we use the stop-gradient function when computing the gradient of $p_\eta(e|x)$ as follows:

$$\nabla_\eta \exp\left(\text{sg}[f_\eta(e)]^T f_\eta(x)\right),$$

meaning that we do not differentiate through the explanation embeddings, but only through the query data point embeddings. In early experiments, we found that this decision contributed to training stability, while improving computational efficiency, and we confirm that we observe no differences in model accuracy as a result.

C.3 Experiment Confidence Intervals

We compute confidence intervals for our synthetic data tasks to represent *seed variance* around some mean seed performance. We represent seed variance in figures rather than sample variance because the sample variance is fairly low with 50,000 test points and could be driven arbitrarily low with more generated test points. For instance, the 95% confidence interval for a model accuracy of 90% would be ± 0.26. To calculate seed variance, we run 10 random seeds for our baseline condition (no-retrieval) with the default synthetic task setup.

D Synthetic Task Generative Process

The required parameters to the data generation include: (1) a training sample size *sample-size* and (2) *num-tasks*, the number of unique integer pairs to be counted, or, equivalently, the number of points per *index*, n_{task}. In all experiments, we use a maximum integer value of 100 to appear in the sequences, and a maximum *index* value of 10,000. We give the general generative process below. Note that the dev and test sets are constructed with the extra constraint that sequences must not appear in the training data. Further note that this is the generic version of generative process, and in some experiments the process is altered. For example, in RQ3, *indicator* is always 1 and the construction of the map from *index* values to (m, n) tuples occurs in a special way described in the experimental design for RQ3.

1. Sample $\{index_t\}_{\tau=1}^{num\text{-}tasks}$ from the uniform distribution over integers {1,...,10000} without replacement.

2. Sample $\{(m, n, r, d)_t\}_{\tau=1}^{num\text{-}tasks}$ from the uniform distribution over integers, $unif([1, 100]^4)$, without replacement and requiring that $m \neq n \neq r \neq d$.

3. Define the set $\{(index, m, n, r, d)_{index})\}$ for *index* and (m, n, r, d) drawn from their respective sets, without replacement, in an arbitrary order.

4. Compute the number of points per *index*, n_{task} = *sample-size // num-tasks*.

5. For each *index* $\in \{index_t\}_{\tau=1}^{num\text{-}tasks}$:

   (a) Sample a vector of length n_{task}, balanced between 1s and 2s, that gives the values of $\{indicator_p\}_{p=1}^{P}$ for the P points with that *index*.

   (b) Sample a vector of length n_{task}, balanced between 0s and 1s, representing whether the features $\mathbb{1}[\#m>\#n]$ and $\mathbb{1}[\#r>\#d]$ should correlate (1 implies they are equal, and 0 unequal). This balance changes when the strong-weak correlation is intended to change.

   (c) Sample a vector of length n_{task}, balanced between 0s and 1s, representing whether (m, n) or (r, d) should be the more *numerous* integers in the sequence (so that there is no bias, even randomly, between features by size).

   (d) For $i \in 1 : n_{task}$:

      i. Place the *index* in the first element of an empty array, and the *indicator* in the second.

      ii. Based on the i^{th} elements of the three vectors described above, allocate samples of the integers in $(m, n, r, d)_{index}$ into the remaining 18 slots.

      iii. If there are any remaining slots after these integers are randomly allocated, fill them with i.i.d. samples from $unif(1, 100)$.