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

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

Abstract

Many methods now exist for conditioning model outputs on task instructions, retrieved documents, and user-provided explanations and feedback. Rather than relying solely on examples of task inputs and outputs, these approaches allow for valuable additional data to be used in modeling with the purpose of improving model correctness and aligning learned models with human priors. Meanwhile, a growing body of evidence suggests that some language models can (1) store a large amount of knowledge in their parameters, and (2) perform inference over tasks in unstructured text to solve new tasks at test time. These results raise the possibility that, for some tasks, humans cannot explain to a model any more about the task than it already knows or could infer on its own. In this paper, we study the circumstances under which explanations of individual data points can (or cannot) improve modeling performance. In order to carefully control important properties of the data and explanations, we introduce a synthetic dataset for experiments, and we also make use of three existing datasets with explanations: e-SNLI, TACRED, and SemEval. We first give a formal framework for the available modeling approaches, in which explanation data can be used as model *inputs*, as *labels*, or as a *prior*. After arguing that the most promising role for explanation data is as model inputs, we propose to use a retrieval-based method and show that it solves our synthetic task with accuracies upwards of 95%, while baselines without explanation data achieve below 65% accuracy. We then identify properties of datasets for which retrieval-based modeling fails, such as having explanations that are not sufficiently relevant *across* data points. With the three existing datasets, we find no improvements from explanation retrieval. Drawing on our findings from our synthetic task, we suggest that at least one of six preconditions for successful modeling fails to hold with these datasets.[1]

1. Introduction

To provide signal for learning, traditional supervised learning algorithms use labels consisting of class IDs or a number in regression settings. Yet training models with data in this form provides the minimum possible supervision for learning a task. Consider how deeply this style of learning contrasts with the way a person can learn a task by getting verbal explanations from someone helping them in addition to just the error signal from their performance. Access to such feedback can accelerate learning, resulting in less error-prone behavior, while also aligning the learned behavior with the teacher's prior on what behaviors are good. Since this sort of training should yield efficient and safe outcomes, the contrast between machine and human learning points to natural question: How can we incorporate natural language *explanations* into learning algorithms?

A long line of past work has sought to use explanations, rationales, instructions, and other similar data to improve 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; Andreas et al., 2018; 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., 2020) 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).

What is surprising about the sheer breadth of approaches in these works is that they all aim to incorporate essentially the same kinds of information. We can describe each of these approaches as trying to augment models with (1) information not available through their inputs or in their parametric knowledge, or (2) a further specification of the task that

[1]Our code and data will be made publicly available at: https://github.com/peterbhase/ExplanationRoles

is informative about which models are good. Improving models in this manner is a natural goal of approaches using explanations, since one purpose of an explanation is to communicate a mental model (Doshi-Velez & Kim, 2017; Miller, 2019). But how do explanations get used as additional labels, as inputs, as regularizers, as structured variables, and as rules for automatic data labeling? Even under a general notion of what an "explanation" is, e.g. the answer to some *why-question* (Miller, 2019), this kind of data plays an impressive number of roles.

Yet there are tasks where explanations do not fulfill these roles effectively, as improvements in performance prove elusive even when thousands of explanations are gathered (Narang et al., 2020; Hase et al., 2020). In fact, there is reason to think that for some tasks models will not need additional information or further task specification of the kind explanations provide. This is because large language models now (1) store a great amount of knowledge in their parameters (Roberts et al., 2020; Lewis et al., 2020), and (2) infer tasks at test time from the input itself (Radford et al., 2019; Brown et al., 2020; Weller et al., 2020). So in some situations we may not be able to explain to a model more about a task or a data point than it already knows or could infer on its own. What remains unclear, however, is the set of conditions which distinguish situations where explanations will be helpful from those where they will not be helpful in practice or cannot be in principle.

In this paper, we (1) give an argument for the role of explanations in modeling that helps us understand how explanations have been used in such distinct ways and points us toward suitable methods, and (2) we experimentally study the conditions under which explanations are or are not helpful to models, using a specially designed synthetic task and three existing datasets with explanations given for individual data points. The modeling approach we ultimately propose is to perform retrieval over past explanations and provide them as inputs to a model at prediction time (see Sec. 2.6), which is the approach we reach following our broader argument in Sec. 2. Our synthetic task (described in Sec. 3) is designed to have analogous properties to existing real (i.e., human-curated) data, and it is especially useful here as it enables us to test a number of hypotheses that we could not test with existing datasets.

Using RoBERTa as a representative large language model (Liu et al., 2019) and Sentence-BERT as a retrieval model (Reimers & Gurevych, 2019), we investigate a number of **primary research questions**, each given with some brief context:

1. **RQ1.** Since models can infer tasks from sequences at test time, providing task information may not be helpful. *When can models solve our synthetic problem by inferring each sequence's task, and when must they be given the task information?*
2. **RQ2.** Explanations seen in the past may help with predicting future data points. *Can retrieval of past explanations enable a model to solve our task?*
3. **RQ3.** Useful information might be distributed over several explanations. *Can models aggregate information across explanations for better prediction?*
4. **RQ4.** We can either let pretrained models combine explanations by giving them all as textual input, or we can pool extracted feature representations. *What is the best way to compute explanation representations for prediction?*
5. **RQ5.** Good explanations pertain to the data point they are given for, but *what makes an explanation relevant across data points? What enables a retrieval model to find relevant explanations for a new data point?*
6. **RQ6.** One intuitive use case for explanations is to encourage models to rely on causal features rather than spurious correlations. *Can explanations help models learn to use strong features rather than weak ones?*
7. **RQ7.** Here, the training signal for a retrieval model depends on how the classifier uses the explanations the initial retrieval model can provide. *How does the co-dependence between classifier and retrieval model influence the viability of joint training?*
8. **RQ8.** After identifying a set of conditions which determine whether retrieval-based modeling can succeed in our synthetic task, we ask: *does retrieval of explanations improve model performance on existing datasets?*

We describe the experimental results for these research questions in Sec. 6, and we discuss their implications in Sec. 7.

2. Formalizing the Role of Explanations in Modeling Data

In what follows we discuss what we mean by the term "explanation" (Sec. 2.1), our formal framework for the uses of explanations in modeling and relevant work on the subject (Sec. 2.2), a unified view of the roles of explanations in modeling (Sec. 2.3), how explanations complement the input in NLP tasks (Sec. 2.4), and the model we use in this paper (Sec. 2.6).

2.1. What Is an Explanation?

The term "explanation" has no consistent definition in machine learning, as methods papers use it in multiple senses and even opinion papers present definitions of limited specificity. For our present purposes, we use the term to refer to the kinds of data one might collect if asking a person to answer the question, "Why does data point x have label

Illustrative Example #1

τ: When asked for travel times, give them in terms of travel by a car.

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

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.

y: About 8 hours.

Illustrative Example #2

τ: What are the names of people in the text?

s: 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.

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.

y: Babbage, Lovelace, Mary Somerville.

Figure 1. Hypothetical data and explanations for illustration. In these examples, s is the kind of input one might expect a model to produce the correct output for after some amount of finetuning on (s, y) pairs. For some models s may be sufficient, while others may benefit from additional information as provided by τ or e.

y?" This is a generic formulation of the explanation as an answer to a *why-question* of the kind discussed in Miller (2019). For a more extensive discussion of explanations in the context of AI, we refer the reader to this work. Rather than try to give a delimiting, formal definition of the kind of data generated from this question, we proceed with some illustrative examples, shown in Fig. 1. In Sec. 5, we describe human explanations used in experiments.

2.2. Formal Framework and Relevant Work

In this section, we lay out our theory of how explanations may be used in modeling a task. With a focus on supervised learning, we characterize the modeling process here in terms of MAP inference over model parameters θ,

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

where Y is a set of labels for inputs X, and the pair constitute a standard supervised learning task. We refer to the role of Y in this probabilistic model as the target, X as an input, and $p(\theta)$ as a prior. Whereas X is intended as the data observed at prediction time, we allow for a latent variable Z to be included as follows:

$$p(\theta|X, Y) \propto \int_{\mathcal{Z}} p(Y|Z, X, \theta)p(Z|X)p(\theta)dZ$$

Both X and Z will be considered as "model inputs" below. Note that we intend this framework to extend to a many-task situation, which we define as a case where several distinct conditional distributions produce the data $\{Y, X\}$. All that is required is that Z indicate the task τ to be solved, i.e. which conditional distribution should be computed.

A few examples: For supervised classification, the task is to map an input X to a label Y, and Z could be a document retrieved from a database. In autoregressive modeling, X and Y are sequences of tokens which may appear in either role depending on the current context and positions to be predicted, and Z could be a textual description of the sequence prediction task or a set of unobserved token positions one intends to marginalize over. Below, we will refer to *tasks* interchangeably with a *function* to be computed and *parameters* of the true model of the data. We mean to index the conditional distribution for each task and refer to the parameterized function that computes it: $p_\tau(y|x) = f_\theta(x)$.

Below we describe existing approaches to using explanations, categorized in our framework. An overview of the corresponding graphical models is shown in Fig. 2.

Using Explanations as Targets. Explanations may be used as additional supervision (i.e. as Y), depending on the ultimate modeling goals (shown as Multi-Task in Fig. 2). For instance, Pruthi et al. (2020) consider the use of attention-weight explanations (from a model) as targets in a multi-task framework, and they find that the explanations make for useful targets in helping another "simulator" model predict the explained model's outputs. Meanwhile, natural language explanations have been treated as targets in a multi-task, sequence-to-sequence framework, using datasets with free-form textual annotations 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 surprising and possibly 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, as measured by task performance.

Using Explanations as Inputs. Using additional model inputs may be valuable for solving some tasks (i.e. additional X or Z). The first family of approaches here uses explanations directly 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 in a reasoning task by including relevant factoids in the text input. In other settings, Co-Reyes et al. (2019) provide online natural language feedback to RL agents, which helps them learn new tasks on the fly, and Rupprecht et al. (2018) take a similar approach to interactive image segmentation with language feedback.

A key question with these approaches is whether it is sensible to collect explanations at prediction time. In an interactive setting, this is reasonable given that human attention is already demanded and system performance is monitored by a human. However, for cases where total automation is a desired outcome, it may not be feasible to collect explanations at test-time. There is also a risk of leaking the label through the additional data. Free-form human explanations tend to directly reveal the label when collected for tasks such

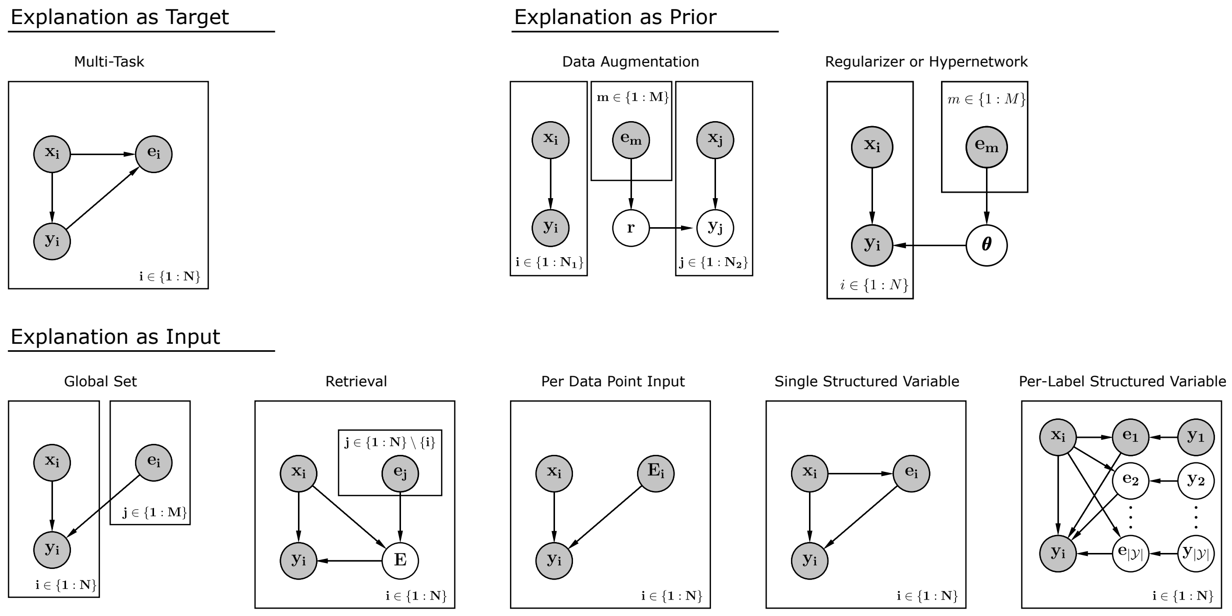


Figure 2. Graphical models for several approaches to using explanations as *targets*, as *inputs*, and as *priors*. Typically past works do not condition on human-given explanations at test time, unless they are collected in an interactive manner with a user or specially designed to not leak the data point label. Note prior works may add or remove dependencies in some cases.

as NLI and QA (Hase et al., 2020; Wiegreffe et al., 2020). Here, what is essentially the cost of human labeling could be mistaken as an improvement in model performance.

There are a few ways to avoid collecting 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 (shown as Global Set in Fig. 2). This can work well for handling up to a hundred or so explanations, but cannot scale to settings with many thousands of explanations. Zhou et al. (2020) treat explanations as latent variables when modeling datasets where only a subset of data points have explanations, and at inference time they retrieve explanations from the training data (Retrieval in Fig. 2.4, with one difference noted here). However, they do not learn the retrieval model, and during training they allow for a data point’s own explanation to be conditioned on as its label is predicted. Since explanations are not available for test data points, this leads to distribution shift between training and test-time inference, and it may introduce label leakage during training predictions. Instead of retrieving explanations, a few works condition on explanations generated at test-time using generative models learned with human explanations as supervision, which are represented as Single Structured Variable and Per-Label Structured Variable in Fig. 2 (Camburu et al., 2018; Rajani et al., 2019; Kumar & Talukdar, 2020; Hase et al., 2020; Wiegreffe et al., 2020). While this form of intermediate supervision could in principle help models learn useful structured variables (the explanations) for prediction, these methods have not produced sustained improvements in model accuracy.

Using Explanations as Priors. Here, we group together any approach to defining or learning a distribution over model parameters, including those that condition on some data, $p(\theta|data)$. We note that this is a prior over model weights not in the sense that the distribution is independent of any data (which it is not), but rather in the sense that the posterior parameters are conditioned on the prior. One natural way to use explanations is to constrain the learned model, e.g. by constraining the conditional distributions the function can express (Srivastava et al., 2017; 2018), or through placing priors over how features are weighted or extracted (Zaidan et al., 2007; Small et al., 2011; Zhang et al., 2016; Bao et al., 2018; Liang et al., 2020; Pruthi et al., 2020). Other works map directly from text to parameters in models (Ba et al., 2015; Andreas et al., 2018), in effect learning a prior $p(\theta|text)$ (though Andreas et al. (2018) condition on generated rather than human-provided text at test-time). 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 augmenting a task with data drawn from some prior distribution $p_\theta(y|x)$ given by the noisy labeling mechanism (show as Data Augmentation in Fig. 2). Critically, in each of these cases, the prior over model weights is some function of explanations, meaning that we require an *interpretation* $\mathcal{I}$, whether learned or given by humans, of how the explanations encode information about the model. We will write that a prior over models is given by an interpretation function on a set of explanations: $p(\theta|\{e\}) = \mathcal{I}(\{e\})$. This kind of function can serve either as a regularizer during training or a hypernetwork that directly outputs model parameters or, equivalently, some task representation (Ha et al., 2017).

2.3. How Explanations Achieve One Goal As Targets, Inputs, *or* Priors

Each of the above methods of supplying information to the modeling process may appear rather distinct, but in principle they can all be used to influence the behavior of a learning algorithm as represented in the posterior parameters. In fact, we observe situations where a single piece of data can be used either as a target, input, or information yielding a prior. Below, we describe a few such situations in simplified terms, providing some justification for how a single format of explanation data might be used as a label, input, or prior. Ultimately, the fact that these various roles can fulfill a single purpose helps us understand how explanations have historically been used with some success in each of the apparently disparate roles. We should note that it was already clear that training better models was one *goal* of using explanations in modeling. We would expect a priori that explanations are suited to this goal given that one underlying purpose to explanation is the communication of a mental model (Doshi-Velez & Kim, 2017; Miller, 2019).

Using Data as a Target *or* Prior. Adopting terminology from Pruthi et al. (2020), we refer to a *teacher* giving explanations to a *student* who is learning a task. Suppose a student is modeling a simple 1-D regression problem ($x \in \mathbb{R}$) as $y \sim \mathcal{N}(y \mid \theta x, \sigma^2)$, for data $D = \{x_i, y_i\}_{i=1}^n$, using a known σ^2 and a Normal prior $p(\theta)$. In this case, the teacher could in principle induce any MAP estimate they wish by adding a single data point (x_1, y') to D, a copy of the first data point with a new label. Of course, the teacher could also induce any desired MAP estimate by directly modifying the student's prior using a particular interpretation function, $p(\theta|y') = \mathcal{I}(y')$. This is simply an illustrative example where one can achieve the same learning outcomes either by providing additional targets or using a particular prior. A more serious analysis would be required to formalize the argument for neural language models and objectives for structured outputs. Thus far, natural language explanations have made no difference to task performance when used as targets (Narang et al., 2020; Hase et al., 2020). The evidence is more favorable for using attention weights from a model as targets, but Pruthi et al. (2020) find this form of explanation to work better as a prior.

Using Data as an Input *or* Prior. Now consider a multivariate regression setting with $y \in \mathbb{R}$ and features $x = (x_1, x_2)$ with $x_1 \in \mathbb{R}$ and $x_2 \in \{0, 1\}$, where the true model is: y is linear in a continuous feature x_0, with the strength of the relationship modulated by the binary feature x_1 (written as $y = \beta_{11}x_1 + x_2 \times \beta_{12}x_1$). Notice that, per our definition of a task above, x_2 is exactly a task representation τ, since it controls for which of multiple functions define a conditional relationship $p(y|x_1)$. Hence, we can treat x_2 as a task representation and define an interpretation $\mathcal{I}$ to give a prior over the weight on x_1, $p(\beta_1|x_2) = \mathcal{I}(x_2)$. A model of this form takes the appearance

$$p(y|x) = p(y|x_1; \beta_1)p(\beta_1|x_2)p(\beta_1) \tag{1}$$

Interestingly, there will exist equally predictive models of the form (1) as there will for a standard regression model,

$$p(y|x) = p(y|x_1, x_2; \beta_{11}, \beta_{12})p(\beta_{11}, \beta_{12}). \tag{2}$$

With the benefit of hindsight, we can say that the simplest interpretation function to represent $p(\beta_1|x_2)$ places a point mass on β_{11} when $x_2 = 0$ and on $\beta_{11} + \beta_{12}$ when $x_2 = 1$. But we could also learn the prior $p(\beta_1|x_2)$, either with direct supervision for β_1, by differentiating through a point estimate $\hat{\beta_1}$, or by marginalizing over a random variable for β_1. In this manner, one can learn equally predictive models treating x_2 as an input to a single learned function or a task representation that carries information about model parameters. As before, this is only a simple example, and a more formal analysis would be required to precisely identify this phenomenon when using textual data with methods that may perform interpretation and prediction within one large computation graph (i.e. existing neural models).

2.4. How Explanations Complement the Input in NLP

The ambiguity between considering data as an input or prior is of great relevance in NLP now as a growing body of evidence suggests that pretraining language models teaches them how to do inference over tasks at test-time. Indeed it appears that sufficiently large language models do "infer and perform the tasks demonstrated in natural language sequences in order to better predict them," as Radford et al. (2019) hoped for. For example, GPT-3 metalearns how to do sequence prediction over the course of pretraining, which equips it for zero-shot prediction given task descriptions and examples (Brown et al., 2020). Even GPT-2 demonstrates the ability to infer the task at prediction time, e.g. for summarization purposes given the "tl;dr" prompt (Radford et al., 2019).

But these results leave open the question of when and to what degree task information is helpful for prediction in well-defined tasks. In question answering, for example, when should we think that inferring or conditioning on task information is helpful, as opposed to relying on a task's "input" alone? In fact, for cases like QA, it is even difficult to identify what counts as a *sufficient* input for the

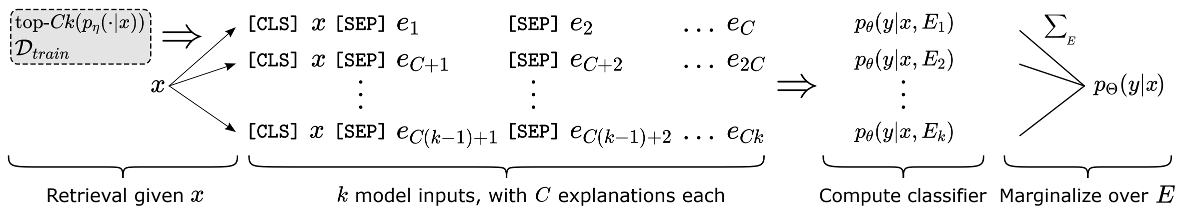

top-$Ck(p_\eta(\cdot|x))$
$\mathcal{D}_{train}$
x
[CLS] x [SEP] e_1 [SEP] e_2 ... e_C
[CLS] x [SEP] e_{C+1} [SEP] e_{C+2} ... e_{2C}
[CLS] x [SEP] $e_{C(k-1)+1}$ [SEP] $e_{C(k-1)+2}$ ... e_{Ck}
$p_\theta(y|x,E_1)$
$p_\theta(y|x,E_2)$
$p_\theta(y|x,E_k)$
$\sum_E$
$p_\Theta(y|x)$
Retrieval given x
k model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

Figure 3. 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.

task to be solvable by some model without additional information clarifying ambiguities in the task or providing relevant background knowledge. Consider that Roberts et al. (2020) use pretrained models to answer questions without any further input, while Lewis et al. (2020) find it helpful to retrieve relevant documents from Wikipedia before answering, drawing a distinction between parametric and non-parametric model memory. Yet when Weller et al. (2020) study how models generalize across tasks when conditioning directly on task descriptions, they formulate the descriptions as *questions* with the task's data given as accompanying documents. **Hence we see one model's input x used as another model's task description τ, and in both situations additional (possibly retrieved) data can improve task performance**.

Our experiments provide some answers to the remaining question of when and degree to what degree task information is helpful, and based on our experiments in Sec. 6, we describe conditions for models (1) inferring tasks from the input alone, (2) benefiting from the retrieval of additional information, and (3) being able to learn the retrieval.

2.5. Assumed Structure of Data

In this paper we assume we have data of the form $D = \{x_i, y_i, e_i\}_{i=1}^n$, where (x, y) is a standard data point in a supervised classification task, and e is some kind of data collected in response to a question like "why does data point x have label y?" In our experiments with both synthetic and human-curated data, x and e are sequences of tokens from a fixed vocabulary. The approaches we present will allow for unexplained training data, meaning some or even most e_i may be missing. The model may use any number of free-floating explanations too, i.e. e_i without corresponding (x_i, y_i) pairs, though this does not apply to datasets in this paper.

2.6. Our Model

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). Given our discussion above, a few reasons point us in this direction: (1) since past explanations may be useful for future predictions, while collecting explanations is costly and can lead to label leakage, we want to avoid collecting explanations at test time; (2) this method may directly condition on relevant information that is useful for reasoning tasks (Talmor et al., 2020); (3) textual data can provide useful task information when serving as a model input, and hence this is a natural way to learn a prior over tasks (Brown et al., 2020; Weller et al., 2020) (4) retrieval is more scalable than conditioning on a global set of explanations, and (5) using explanations as structured variables and as targets do not appear to be promising approaches at the moment (Hase et al., 2020; Wiegreffe et al., 2020; Pruthi et al., 2020).

So, we use a retrieval-based model that treats retrieved explanations as latent variables to be marginalized over. 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. 4. 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

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

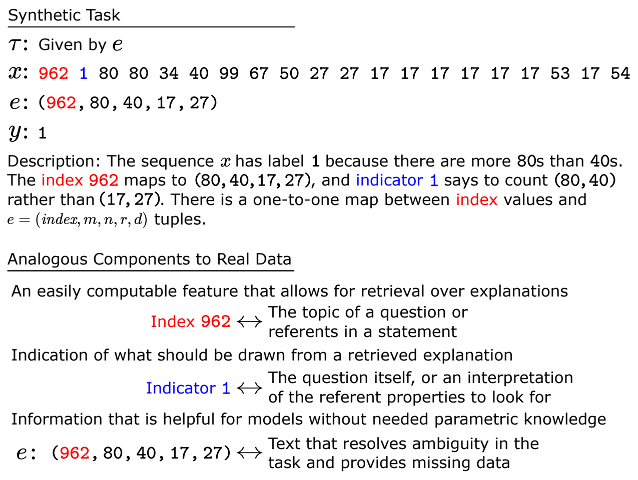
where (1) a *context size* C will control the size of the explanation set, (2) a value of k implies that the top Ck will be retrieved, and (3) we sort these Ck explanations into sets in order of their probability $p_\eta(e|x)$.

We represent the overall approach in Fig. 3 for one method of computing $p_\theta(y|x,E)$ (described fully in Sec. 4), where explanations are concatenated with the query sequence. Flowing from left to right, Fig. 3 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. We study how the "relevance" of explanations to other data points influences task solvability through experiments in Sec. 6.5. 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.

3. Synthetic Task

We design a synthetic dataset so that we can carefully control several important properties of the data, though we also make use of several human-curated datasets (described in Sec. 5). Designing a synthetic dataset for the task at hand is a useful exercise for a number of reasons. At a high level, it helps us formalize our intuitions regarding what makes the task solvable or not solvable given (1) certain inputs, (2) certain modeling approaches, and (3) certain available explanations. A critical part of this procedure is that, as we do so, we make *disputable decisions* regarding how the synthetic task maps back onto reality. When all is said and done, one can ask if the properties of the proposed data and modeling paradigm do in fact reflect how we expect modeling will work with human-given, natural language explanations. In this spirit, we claim that our synthetic task shares a few important properties with human-curated data, which are described in Sec. 3.2. Lastly, as a practical matter, it allows us to study how various properties of the data allow for successful modeling with existing methods. In this paper, we are able to provide experimental answers to six of our eight primary research questions only through synthetic data, and not with available datasets. Hence, we introduce a synthetic task for our present purpose. For further discussion of the pros and cons of synthetic datasets, see (Liu et al., 2021). In Fig. 4, we show an example data point, along with a description of how it gets its label. The premise of our task is to classify sequences using counts of different integers in the sequences. The basic idea of counting integers is drawn from De Cao et al. (2020). They propose a toy task requiring a model to count whether there are more 8s than 1s in a sequence, with the purpose of evaluating a model interpretation method.


Synthetic Task

τ: Given by e

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

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

y: 1

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). There is a one-to-one map between index values and $e = (index, m, n, r, d)$ tuples.

Analogous Components to Real Data

An easily computable feature that allows for retrieval over explanations

Index 962 ⟷ The topic of a question or referents in a statement

Indication of what should be drawn from a retrieved explanation

Indicator 1 ⟷ The question itself, or an interpretation of the referent properties to look for

Information that is helpful for models without needed parametric knowledge

e: (962, 80, 40, 17, 27) ⟷ Text that resolves ambiguity in the task and provides missing data

Figure 4. Examples of our synthetic task and analogies we draw to human-curated existing data.

3.1. Generated Data

We wish to design a task where, while it would be possible to solve the task by learning a function $y = f(x)$, it would be easier if you could condition on relevant explanations and learn $y = f(x, e)$. We propose a few task variants, but the core of the task is that, given a sequence x, the binary label will be determined by whether there are more of an integer m in the sequence than there are of an integer n. We assign a one-to-one map between the integers (a, b) to be counted and a set of special integers each sequence includes as its first two elements, which we term the *index* and *indicator*. For our purposes, a key property of this kind of task is that a model could succeed by memorizing the map between (*index*, *indicator*) and the integers it needs to to count. However, it should be much easier to solve the task when directly conditioning on those integers, i.e. learning a function from (x, a, b) to y. Here, the "explanation" (a, b) is a plausible answer to the question of why data point x has label y, because this information determines the feature that causes the label.

Rather than just using the *index* to map to the two numbers that need to be counted, we include the *indicator* so that models can succeed by integrating information from x and e. An explanation is given as (*index*$, m, n, r, d)$, where either (m, n) or (r, d) is the integer pair that actually needs to be counted. The opposite pair will be a distractor feature whose relative counts match those of the causal feature 50% of the time. Then the *index* will map to $(m, n, r, d)_{index}$, and the *indicator*, either a 1 or a 2, will tell whether it is the first integer pair in the explanation (m, n) or the second (r, d) that needs to be counted (as displayed in Fig. 4). As a result, with *num-tasks* many *index* values, there will be $2 \times$ *num-tasks* possible pairs of integers that have to be counted. In general, we will refer to a sequence's *task* as the function that counts the relevant integers for that particular sequence, meaning we view our dataset to be composed of many (similar) tasks, each well-defined for a set of sequences.

We next describe the exact dataset in detail. The full generative process is given in Appendix C. We give typical values that dataset parameters take on, and in Sec. 6, we note differences from this default setting as they become relevant to each experiment. The resulting data is:

1. Train set: 5000 sequences of 20 integers (including *index* and *indicator*), where there are 500 unique values of *index* in the dataset drawn from $unif(1, 10000)$. **For each *index*, there are 10 distinct x_i that share a common explanation**, (*index*, $m, n, r, d)_{index}$. The values of $m, n, r,$ and d are drawn from $unif(1, 100)$ while filtering samples s.t. $m{\neq}n{\neq}r{\neq}d$. The corresponding 10 values of *indicator* are balanced between 1 and 2. Half of the points have label $y{=}1$, meaning that either $m{>}n$ or $r{>}d$, depending on which feature is causal. Half the time, the non-causal integer pair in (m, n, r, d) (i.e., the one not indicated by *indicator*), has counts with the same rank-ordering as the causal feature's counts. In each x_i, after $m, n, r,$ and d have been randomly placed into the sequence, any 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.

3.2. Important Data Properties

Analogous Properties to Human-Curated Data. We claim that aspects of our synthetic task are analogous to properties real (i.e. existing, human-curated) data might take on. We first highlight a few properties of the Illustrative Examples in Fig. 1. Here, s is the kind of input for which one might expect a model to produce the correct output after some amount of finetuning on an appropriate dataset, while τ offers explicit task instructions and e is an explanation of the data point's label. We expect that, for some models, τ and e will provide useful additional information for the task that is not represented in s or is difficult to infer from s. Models might more easily extract this information from τ or e than they can infer it from s, allowing for better task performance. However, a model may infer any "hidden" information perfectly well without relying on these variables, especially after some amount of finetuning on (s, y) pairs. Without finetuning, a model may already be pretrained to interpret task instructions (Brown et al., 2020), or the model may already know the hidden information (Roberts et al., 2020), meaning the knowledge encoded is in their parameters and accessible in the right circumstances.

Now, regarding our synthetic data, we first claim that e is an explanation in the sense that it is a plausible answer to the question, "why does point x have label y?" The explanation gives the information which determines 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 a question or the referents of a statement (the things referred to): both are computable features that make retrieval-based modeling possible. Likewise, good models will combine the *indicator* and explanation to identify the causal feature in the same way that a good QA model would figure out what to look for in a document by first understanding what the question asks for or the referent properties it should be looking for.

Our task shares another important characteristic with human-curated data: whenever retrieval could be helpful, models can learn to directly infer the hidden information from the input alone. In the synthetic task, this looks like learning the function from the *index* to the integers to be counted. With question answering, for example, a model could learn the map between a certain topic and the set of facts that could be needed to answer questions about that topic. This may be harder than learning a retrieval model for a given dataset, but it is possible in theory and would render the additional data for retrieval irrelevant. In our experiments in Sec 6.1, we outline situations where this map is learned by models, making retrieval unnecessary.

Data Parameters, *Relevance*, and Strong Features. There are a few parameters to the data generation that heavily shape our expectations of the task's solvability. The first is the number of unique values of *index*, which we will refer to as the number of tasks, *num-tasks*. With a fixed training set size, *num-tasks* determines the number of data points per task, n_{task}. For example, while we will typically have 10 points per task, decreasing the number of tasks to 100 would mean there would be 50 points per task (with 5000 training points). This is a particularly important property because it determines how explanations will be *relevant* across data points. Here, we define an explanation for one data point e_i to be *relevant* to another sequence s_j when e_i is informative about what sequence s_j's task τ_j is. Recall that by τ_j we refer to the function counting the integers $(a, b)_j$. Formally we will say that a *relevance function* on s and e yields some distribution over the task parameters:

$$p((a,b)_j|s_j, e_i) = f(s_j, e_i).$$

In the standard version of our synthetic task, one such relevance function could place all probability mass on (m, n) if $indicator = 1$ and the *index* in s_j and e_i matched (or (r, d) if $indicator = 2$). If the *index* does not match, then there would be no information about what τ_j is, since we randomly sample *index* and (m, n, r, d) values when pairing them. To obtain a smoother, more continuous level of relevance between sequences and explanations, we can also define a predictable relationship between index_i and $(m, n, r, d)_i$ so that $(a, b)_i$ and $(a, b)_j$ are close together (under some distance metric) whenever index_i and index_j are close together. We describe experiments comparing the two settings of binary and smooth relevance in Sec. 6.5.

Next, note that we can vary the degree to which the non-causal feature is correlated with the causal (strong) feature. In the case of perfect correlation, we have that $\#m>\#n$ iff $\#r>\#d$ and $\#m<\#n$ iff $\#r<\#d$, regardless of which is the causal feature. This allows us to test whether explanations can induce models to rely on causal rather than non-causal (weak) features. While this is an intuitive reason for thinking explanations should be helpful for models, we show in Sec. 6.6 that models can correctly use explanations for selection between correlated features only in a narrow set of situations.

Finally, *index* can be removed from each sequence to more closely imitate a situation requiring task inference. While in principle models can learn the map from *(index, indicator)* to (a, b), in fact we find that models will infer the task even when *index* is removed from the sequence (Sec. 6.1). Ostensibly they do so by counting the sequence integers: those which appear often are likely to make up (m, n, r, d).

3.3. Kinds of Explanations

The data we have described so far includes only a single form of explanation, $e = (\mathit{index}, m, n, r, d)$, which we will call our *full-info* condition. As long as a retrieval model returns relevant explanations, the task for a sequence can be read off from this kind of explanation. Yet, rather than giving a full description of the task, explanations in existing datasets tend to only partially specify a task or give just a piece of the hidden information for a data point, especially when annotators limit the length of their explanations to a single sentence (Camburu et al., 2018; Wang et al., 2019b).

This leads us to suggest two alternative forms of explanation in our synthetic task, which we refer to as *evidential* and *recomposable* explanations. Given an *index*, **evidential** explanations are generated by adding independent, zero-mean noise to each element in the true $(m, n, r, d)_{\mathit{index}}$, s.t. taking the average across a set of evidential explanations converges in the limit to the true $(m, n, r, d)_{\mathit{index}}$. In our experiments, we will add some noise ϵ drawn from the uniform discrete distribution from -2 to 2.

The second explanation kind, **recomposable**, is designed so that one could infer the task if one had all the relevant explanations for a particular *index*. We create such a situation by breaking the (m, n, r, d) into parts that neatly recompose back into the true set of numbers. Principally, we do so by dividing the explanation into two pieces, $(m, 0, r, 0)$ and $(0, n, 0, d)$, where some points with that *index* have one explanation, and other points have the other. We ensure that both pieces of an explanation appear at least once among the data points for each *index*. We also experiment with a similar setting where we decompose explanations into four pieces, but do we not include results for this condition as we find them to be quite similar to the two-piece setting.

4. Computational Methods

In this section we describe the methods used to compute $p_\theta(y|x, E)$ and $p_\eta(e|x)$ (see Sec. 2.6 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 & Gurevych, 2019).

4.1. Conditioning Mechanisms

TEXTCAT. Represented in Figure 3, 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. We compare the two methods in Sec. 6.4.

4.2. 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 & Gurevych, 2019) as our pretrained f_η, rather than simply a pretrained RoBERTa model (discussed in Sec. 6.7). 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.

	Sample Size				
Dataset	Explns	Train	Dev	Test	$\lvert\mathcal{Y}\rvert$
Synthetic	5000	5000	10000	50000	2
e-SNLI	549,367	549,367	9842	9824	3
SemEval	203	7016	800	2715	19
TACRED	169	68,124	22,631	15,509	42

Table 1. Statistics for Datasets.

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.

5. Experimental Setup

Here, we detail the datasets and important model training details used in our experiments.

Datasets. The standard version of our synthetic task used in experiments is described in Sec. 3. We include experiments with three other (English) datasets. The first, e-SNLI, is the SNLI dataset annotated with human explanations (Bowman et al., 2015; Camburu et al., 2018). The next two, SemEval and TACRED (Hendrickx et al., 2010; Zhang et al., 2017), are relation extraction tasks with a subset of data points annotated by Wang et al. (2019b). Summary statistics from the three datasets are shown in Table 1. For additional details including data preprocessing see Appendix B.

Model Training. We train all models in an end-to-end manner using AdamW with a standard cross-entropy loss (Loshchilov & Hutter, 2017). This would be straightforward given the model's end-to-end structure, except for the fact that with after every gradient update, *all* training explanation representations need to be recomputed in order for future predictions and gradients to reflect the new parameters. Prior work using retrieval models has either periodically updated the document representations (Guu et al., 2020) or left them fixed and only updated the query embeddings (Lewis et al.,

e-SNLI	
x :	*Premise:* After playing with her other toys, the baby decides that the guitar seems fun to play with as well. *Hypothesis:* A blonde baby.
y :	Neutral
e :	Not all babies are blonde.
SemEval	
x :	The SUBJ originates from an OBJ which transcends the speaker.
y :	Entity-Origin
e :	The phrase "originates from an" occurs between SUBJ and OBJ and there are no more than four words between SUBJ and OBJ and OBJ follows SUBJ.
TACRED	
x :	SUBJ's husband OBJ died in 1995.
y :	Person-Spouse
e :	Between SUBJ and OBJ the phrase "'s husband" occurs and there are no more than five words between SUBJ and OBJ.

Table 2. Example data points from the three existing datasets. More examples can be found in Table 4.

2020). We find it is important to update all embeddings at least every epoch, and unless otherwise noted we rebuild the embeddings every 20% of each epoch (see Appendix A for further discussion).

We give important hyperparameters such as the context size C and retrieval parameter k in each experiment description in Sec. 6. We provide an analysis of the influence of hyperparameters on training in Appendix A, but usually we observe that larger values of C and k yield higher accuracies with more stable training behavior. Other hyperparameters for training are also given in Appendix A.

Model Selection and Hypothesis Testing. We report and visualize results on our synthetic dataset with confidence intervals representing *seed variance*, which accounts for variability across sampled datasets and random model training behavior. We do not estimate sample variance because it is quite small using a test set of 50,000 points, with a 95% confidence interval of e.g. ± 0.26 for a model accuracy of 90%. Seed variance is estimated from 5-10 random seeds, depending on the condition. See Appendix B for further details of seed variance estimation. In synthetic data experiments, we comment on effects far larger than the confidence intervals and do not conduct hypothesis tests.

With the three existing datasets, for the majority of conditions, we run three model seeds and select the best model by dev set accuracy. We run only one seed for conditions using the full TACRED training set and the e-SNLI dataset with at least 50,000 training points. With the selected model, we conduct hypothesis tests for a difference in binomial means to check for differences in test set accuracy.

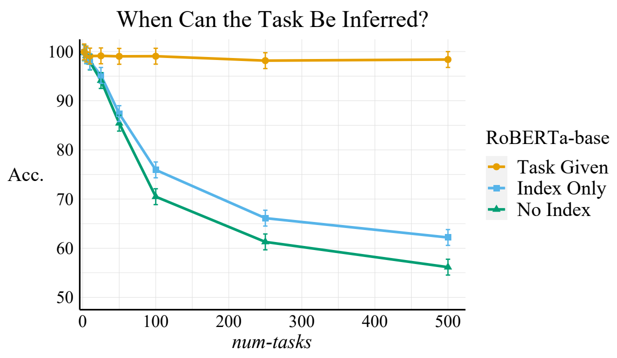


Figure 5. (**RQ1**) Synthetic task accuracy as a function of *num-tasks*.

6. Experiment Design and Results

Below, we give the experimental design and results for each research question in Sec. 1. The first seven research questions are best answered with our synthetic task, and so they each make use of synthetic data (introduced in Sec. 3). See Sec. 6.8 for results with the three existing datasets.

6.1. RQ1: When can models solve our synthetic problem by inferring each sequence's task, and when must they be given the task information?

Design. We measure test accuracy as a function of the *num-tasks* parameter across three conditions. The conditions vary in how task information is available in the input: (1) *task given*, where each sequence has its true task information (m, n, r, d) appended to it; (2) *task signalled*, meaning *index* is given and hence the model can learn the map $index \rightarrow (m, n, r, d)$; (3) *task inferred*, where *index* is not given, so the model must infer the task from the sequence's contents alone. To see the interaction between these conditions and model capacity, we test with both RoBERTa-base and RoBERTa-large, and we also measure the effect of increasing the training set size. Note that, with a fixed training set size, *num-tasks* directly implies the number of points per task, n_{task}. In this experiment, *num-tasks*$= \{2, 5, 10, 25, 100, 250, 500\} \Rightarrow n_{task} = \{2500, 1000, 500, 200, 50, 20, 10\}$.

Results. We show the results in Fig. 5. We see that, when the numbers of tasks is small, RoBERTa-base can infer the task for each sequence and achieve as high an accuracy as if it had been given task information. Yet, **the feasibility of task inference quickly falls off as the number of tasks increases** (equivalent to the number of points per task decreasing), reaching accuracies as low as 62.2% at *num-tasks*= 500. Meanwhile, we observe that providing the *index* does slightly ease the task inference, but the models can by no means memorize the map from *index* to the task information. Regarding model capacity, we find that using

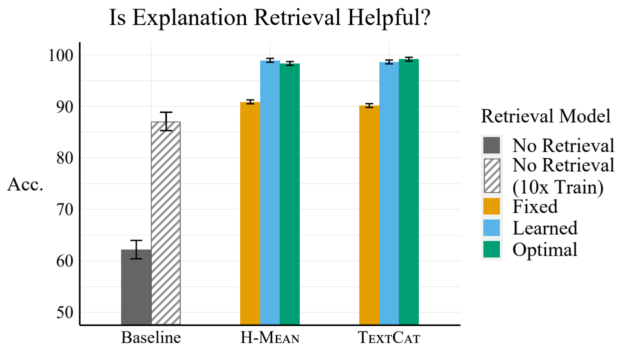


Figure 6. (**RQ2**) Synthetic task accuracy by the conditioning mechanism and retrieval model status, for data with *num-tasks* $= 500$.

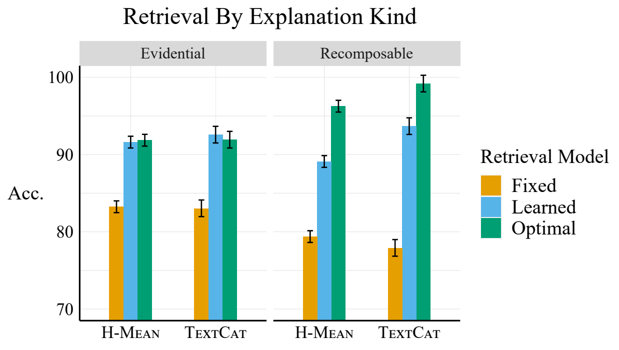


Figure 7. (**RQ3**) Synthetic task accuracy with *evidential* and *recomposable* explanations, grouped by the conditioning mechanism and status of retrieval model.

RoBERTa-large increases model accuracy when the number of *num-tasks* is relatively low (less than 250), but after this point RoBERTa-base performs better (see Fig. 13 in Appendix B). Lastly, we see that increasing the training set size can greatly improve model performance even with *num-tasks*=500, reaching 87.11% with 50,000 training points (trend shown in Fig. 14 in Appendix B). However, we will see in the next section that, in terms of improving model accuracy, even this 10x increase in training size is less effective than using retrieved explanations with 5000 training points.

6.2. RQ2: Can retrieval of past explanations enable a model to solve our task?

Design. Using the *full-info* explanations and data with *num-tasks*$= 500$, 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-to-end, and (3) *optimal* where the optimal retrieval model is used and the classifier is trained. Note that we know the optimal retrieval model assigns the highest probabilities to explanations with $\textit{index}_e$ matching the query point's $\textit{index}_x$, so by using a retriever $p(e_i|x_i) = \exp\left(\mathbb{1}[\textit{index}_e = \textit{index}_x]\right)$ and a context size lower than $n_{\textit{task}}$, we can ensure the retrieved explanations are always relevant. There are two conditions for the conditioning mechanism used: (1) TextCat with C=k=6, and (2) H-Mean with C=4 and k=4, which approximately matches the computational cost of the TextCat condition.

Results. Shown in Fig. 6, the results show that retrieval with Sentence-BERT reaches accuracies above 98%, improving model accuracy by around 29 percentage points over a no-retrieval baseline. Each conditioning mechanism sees roughly the same improvement. Additionally, we can learn a retrieval model that does nearly as well as the optimal retrieval model, improving over the *fixed* condition by about 7 points. Thus, **retrieval of explanations allows the model to perform much better than a no-retrieval baseline**. We see a large improvement in performance from retrieval even when the baseline could learn to infer the task information directly from the *index* value in each input. In fact, explanation retrieval outperforms a no-retrieval baseline with as many as 50,000 training data points (a 10x increase), which obtains 87.11% accuracy.

6.3. RQ3: Can models aggregate information across explanations for better prediction?

Design. We run the same experiment design as for RQ2, using *evidential* and *recomposable* explanations (see Sec. 3.3). With evidential explanations, we shift each integer in the explanation (excluding the *index*) independently by zero-mean, discrete noise $\epsilon \sim unif(-2, 2)$. In the recomposable setting, for each *index* two explanations combine to give the full task information. As in RQ1, we show results here for values of C=k=6 for TextCat and C=k=4 for H-Mean.

Results. We display the results in Fig. 7. First, we observe that for evidential explanations, learned retrieval is close to the optimal retrieval, and the conditioning mechanisms perform very similarly. Yet the models cannot interpret evidential explanations as well as full-info, seeing as even with optimal retrieval both TextCat and H-Mean obtain around 92% accuracy compared to full-info's 98%.

With recomposable explanations, meanwhile, we notice two differences. First, we find that with optimal retrieval TextCat can interpret the recomposable explanations as well as full-info, achieving upwards of 98% accuracy. Yet we observe that learned retrieval falls 6-8 points short of optimal retrieval (depending on the conditioning mechanism). There is no clear reason why this should be, though we can attribute it to the differences in explanations alone. Second, TextCat with learned or optimal retrieval outperforms H-Mean with the same retrieval (by 4.58 points for Learned).

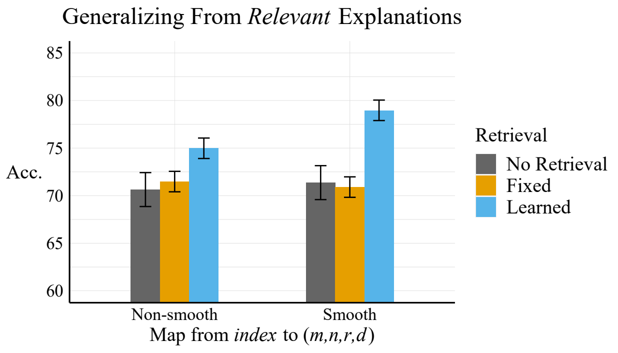


Figure 8. (**RQ5**) Task accuracy with by retrieval model and the *smoothness* of the *index* $\to (m, n, r, d)$ map, shown for 1 and 4 point per task (emphindex). At test time new *index* values are used, meaning models must generalize based on retrieved explanations with similar but never exactly correct (m, n, r, d) values.

We discuss this further in the next section.

6.4. RQ4: What is the best way to compute explanation representations for prediction?

Design. Here we rely on results from the experiments for RQ3, and we also test method performance across training set sizes in $\{1000, 1500, 2500, 5000, 10000\}$, using optimal retrieval with C=5 and k=1 for both TEXTCAT and H-MEAN. Lastly, we consider training time as a relevant factor.

Results. As shown in Fig. 7, with learned retrieval TEXTCAT outperforms H-MEAN by 4.58 points when explanations are broken down into parts that can be recombined to obtain the exact task information. This is especially important as explanations for existing natural language data can give facts and task specifications that may be combined to form a fuller picture of the problem. Additionally, we find that for small sample sizes, TEXTCAT achieves higher accuracy than H-MEAN, by 9.3 points for $n = 1000$ and 9.2 points for $n = 1500$, though the gap shrinks to 1.3 points at $n = 2500$ and the methods perform equally well after $n = 5000$ (see Fig. 15 in Appendix B). As a final consideration, we note that at C=k=4, H-MEAN takes 61% longer to train than TEXTCAT due to the additional model forward passes. So, given favorable performance with recomposable explanations and low sample sizes, as well as the training speed, **TEXTCAT appears to be the preferable conditioning mechanism to H-MEAN**, and unless otherwise stated we use TEXTCAT in experiments henceforth.

6.5. RQ5: What makes an explanation relevant across data points? What enables a retrieval model to find relevant explanations for a new data point?

Design. For retrieval-based modeling to be successful, explanations for one data point must be *relevant* to predicting other data points' labels. So far, we have used n_{task}=10 points per *index*, with test *index* values that have been seen during training, meaning that both during training and testing, "exactly correct" explanations are available for retrieval (i.e. explanations with the true (m, n, r, d) for the data point at hand). To see what is required for explanations to be relevant across data points, we set n_{task} to 1, making every explanation n the train set unique, and we use test data with *index* values not seen in training. As a result of these changes, at both training and test time there are no exactly correct explanations available for retrieval (since we do not retrieve the data point's own explanation). In addition, we restrict the causal feature to always be (m, n), rather than (r, d), for reasons that will become apparent momentarily. To succeed in this setting, models must generalize from explanations given for one data point to a data point with a similar but not identical set of integers to be counted. By default, *index* and (m, n) values are randomly matched, meaning one cannot infer that the explanations are similar for two *index* values given that the *index* values are similar. In our *smooth* condition, we enforce a constraint in data generation so that the *index* and (m, n) values are ordered together, and similar *index* values will have similar (m, n) tuples (see Appendix B for further details).

We also measure the importance of including the *index* in x, which is the easily computable feature linking query data points and explanations. Here, we use the default task setup, identical to that in RQ2, and we learn the retrieval model while using TEXTCAT.

Results. We show results across n_{task} and *smoothness* in Fig. 8. The notable trend here is that learned retrieval clearly outperforms the baseline in the *smooth* condition (by 7.6 points), while it only slightly outperforms the baseline in the *non-smooth* condition (by 4.3 points). In terms of improvement over the fixed retriever, the differences are 8.1 points in the *smooth* condition and 3.5 points in the *non-smooth* condition. This result suggests that **learning to retrieve explanations will be particularly helpful when there is a sufficiently smooth notion of *relevance* between data points and explanations**. The mechanism for this improvement is that, by retrieving explanations with similar *index* values to the data point at hand, a model can guess the task parameters for the current data point since they will be close to the (m, n) values in the retrieved explanations (fitting the definition of relevance in Sec. 3.2).

Regarding the importance of the *index*, we find that **for learning the retrieval to be possible, it is crucial that data and explanations are linked by an easily computable feature** such as the *index*. Without including the *index* in x, learned retrieval accuracy falls drastically from 98.6% to 54.7%.

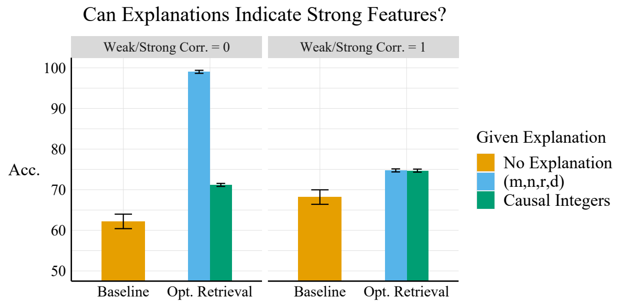


Figure 9. (**RQ6**) Synthetic task accuracy grouped by the explanation kind and correlation between strong (causal) and weak (non-causal) features. In the Causal Integers condition, the model is always given the true pair of integers that must be counted.

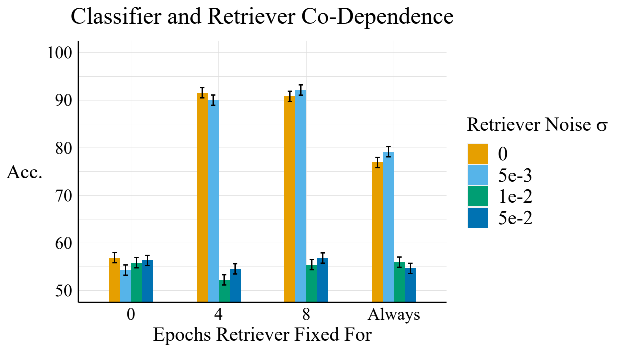


Figure 10. (**RQ7**) The retrieval model must be fixed for some number of epochs for training to succeed. Meanwhile, degrading the quality of the initial retrieval by some amount of random noise can quickly render retrieval unlearnable.

6.6. RQ6: Can explanations help models learn to use strong (causal, generalizable) features rather than weak ones?

Design. One especially intuitive use case for explanations is to help a model distinguish between strong, causal features and weak, spurious features. In this experiment, we vary the correlation between the strong and weak features in the training data along with the kinds of explanations that are retrieved by an optimal retrieval model. Recall that the strong feature in our task is $\mathbb{1}[\#m>\#n]$ when *indicator* $= 1$ and $\mathbb{1}[\#r>\#d]$ when *indicator* $= 2$, while the weak feature is drawn from the opposite integer pair's counts (refer to Sec. 3). We emphasize that our strong and weak features are equally difficult to extract from the input; they differ only in that the strong feature causes the label, and the weak one does not. The explanations either match the familiar form, including all integers $(m, n, r, d)_{index}$, or are restricted to include only the causal integers, (m, n) if *indicator*$=1$ and (r, d) otherwise. When the strong-weak feature correlation is 1, $m>n$ iff $r>d$ and $m<n$ iff $r<d$. When it is 0, the non-causal feature's relative count, i.e. $\mathbb{1}[\#r>\#d]$ if *indicator*$=1$, matches the strong feature's relative counts precisely half the time.

In all of these settings, the dev and test data are unaffected, meaning that a model with high test accuracy must have learned to use only the causal feature. We give additional results with the correlation varied between 0 and 1 in Fig. 16 in Appendix. B.

Results. We see in Fig. 15 that, surprisingly, the only successful situation is when the original (m, n, r, d) explanations are given and the strong-weak correlation is 0, under which the test accuracy is above 99%. Note that, in the other settings, models most likely achieve around 75% accuracy by predicting 1 when $\mathbb{1}[\#m>\#n] \vee \mathbb{1}[\#r>\#d]$, since this strategy yields a test accuracy of 75%.[2] That is, **our "causal feature" explanation fails to help when the strong and features are correlated and even when they are not**. This is surprising because we might expect that, when the correlation is 0, giving the causal feature should allow the model to succeed. After all, we may feel that we are effectively telling the model to count those two integers in every sequence. But **we risk anthropomorphizing the model whenever we suppose its interpretation matches our own.** From the model's standpoint, it sees a sequence of numbers whose *relative counts are always unaffected by the two integers concatenated to the end of the sequence.* Our "explanations" blend in seamlessly with the remainder of the sequence, except for the `[SEP]` token that happens to separate them. Hence we should not be so surprised that the model cannot use these explanations to pick out the causal feature; in fact, it may even be more impressive that the model *does* succeed when the *full-info* explanations are given. Evidently, the model learns a near-perfect interpretation of *full-info* explanations with 5000 training examples.

6.7. RQ7: How does the co-dependence between classifier and retrieval model influence the viability of joint training?

Design. Since the learning signal for the retrieval model comes through the classifier, while the classifier relies on the retrieved explanations for its predictions, there is some co-dependence in their training dynamics. We further measure this co-dependence in a 4×4 design using evidential explanations with $\epsilon = 2$. On one axis, we vary the number of training epochs for which only the classifier is trained and not the retrieval model, in values of $\{0, 4, 8, \infty\}$. On

[2]Half of the time in the test data, the relative counts of (m, n) and (r, d) will agree by chance, meaning that predicting $\mathbb{1}[\#m>\#n] \vee \mathbb{1}[\#r>\#d]$ will yield 100% accuracy. The other half of the time, the features will disagree, and this strategy yields 50% accuracy. The overall test accuracy is then 75%.

the other axis, we degrade the retrieval model by adding i.i.d. Normal noise to every parameter in the model, using σ values of $\{0, 5\text{e}{-3}, 1\text{e}{-3}, 5\text{e}{-2}\}$. To see the effect of choice of Sentence-BERT model, we perform another 3×2 experiment. Using either a randomly re-initialized RoBERTa-base, a standard pretrained RoBERTa-base, or the Sentence-RoBERTa-base model, we evaluate the performance with learned retrieval relative to fixed retrieval.

Results. We show results for the first experiment in Fig. 10. We find that the classifier must be warmed up, or, conversely, the retrieval model must be fixed, for some number of epochs to attain optimal performance. Jointly training both models from epoch 0 on results in failed training runs. Meanwhile, adding noise to the initial retrieval model can quickly degrade its performance and render the retrieval unlearnable. Hence, **both retrieval model and classifier must reach some initial quality before training the other in order for joint training to succeed**.

As for the choice of pretrained retrieval model, we observe that retrieval is learnable only with Sentence-RoBERTa; retrieval is not learnable using RoBERTa-base, which performs about as poorly as using a randomly initialized retrieval model. These results are shown in Fig. 18 in Appendix B.

6.8. RQ8: Does retrieval of explanations improve model performance on existing datasets?

Design. We test the retrieval-based model with three existing datasets: e-SNLI, TACRED, and SemEval. We also vary the training set size between values in {5000, 10000, 50000}, depending on the dataset, since the helpfulness of explanation retrieval could vary by the amount of available training data. Because TEXTCAT achieves favorable results in our synthetic experiments, we use it as the conditioning mechanism here. Within each dataset, we tune C and k between values with the same product Ck, with the exception of e-SNLI using the full train set. For e-SNLI conditions with $n \leq 50000$, we select $(C{=}2, k{=}8)$. We use $(C{=}2, k{=}4)$ for e-SNLI with the full train, given the expense of training retrieval in this setting. For most relation extraction settings we select $(C{=}2, k{=}4)$. See Appendix A for further details.

Unlike in the synthetic data experiments, we consider adding x_j and y_j to the query data point along with retrieved explanation e_j, since explanations might best be interpreted in the context of the data they were given for. In tuning experiments we do not find any evidence for or against adding this extra information (see Table 5 in Appendix A). Here, we do add a textual representation of y_j to the input x_i along with retrieved explanations for relation extraction tasks, since these tasks have a higher number of classes. For e-SNLI, where y_j can be easily inferred from the structure of explanations, we add only retrieved the explanations.

Condition	Model	Acc.	Effect Size
e-SNLI			
n=5000	RoBERTa	84.83 (0.71)	
	TEXTCAT	85.04 (0.71)	0.21 (1.00)
n=10,000	RoBERTa	85.52 (0.70)	
	TEXTCAT	86.03 (0.69)	0.51 (0.98)
n=50,000	RoBERTa	87.90 (0.64)	
	TEXTCAT	87.55 (0.65)	−0.35 (0.92)
n=*full*	RoBERTa	91.06 (0.56)	
	TEXTCAT	91.41 (0.55)	0.35 (0.79)
SemEval			
n=5000	RoBERTa	75.21 (1.62)	
	TEXTCAT	75.73 (1.61)	0.52 (2.29)
n=*full*	RoBERTa	76.94 (1.58)	
	TEXTCAT	76.91 (1.59)	−0.03 (2.24)
TACRED			
n=5000	RoBERTa	84.24 (0.57)	
	TEXTCAT	84.51 (0.57)	0.28 (0.81)
n=10,000	RoBERTa	85.51 (0.55)	
	TEXTCAT	86.14 (0.54)	0.63 (0.78)
n=*full*	RoBERTa	88.29 (0.51)	
	TEXTCAT	88.59 (0.50)	0.30 (0.71)

Table 3. Model accuracies for each dataset across training set sizes (n), with 95% confidence intervals given in parentheses. We do not find retrieval of explanations to improve over baselines for any dataset and training set size.

Finally, for TACRED and SemEval, we compare to the ELV-M method in Zhou et al. (2020), which is H-MEAN with $(C{=}10, k{=}1)$ and fixed retrieval (discussed in Appendix B).

Results. Shown in Table 3, **we see no statistically significant improvements from using explanation retrieval with any combination of dataset and training set size**. Across conditions, the effect sizes are slightly positive on average, but we are unable to assert any particular effect is positive. We also measure how accuracy varies across values of k for finetuned models, but we do not find that increasing k at test time improves accuracy (see Fig. 19 in Appendix B). In fact, the only statistically significant effect we see is from increasing the training set size. For example, doubling the TACRED training data from 5000 to 10000, increases the baseline accuracy by 1.28 (p=.0017).

Yet since we find that retrieval-based modeling succeeds in certain synthetic conditions, there must be a reason that the model fails to work well with datasets such as these. Using the results from this section, we speculate on the possible causes of this failure in Section 7 below.

7. When Can Explanations Help?

In this section we take a position, based on our experimental findings, regarding the possible causes of the success of explanation retrieval in our synthetic task and its failure with e-SNLI, TACRED, and SemEval.

Summarizing our experimental results, we suggest that **in principle, explanations can be helpful for modeling a task when:**

(1) The model can better infer relevant latent information given x and the explanation, relative to using x alone. Relevant latent information includes, for example, pertinent facts and task specifications that can assist with prediction. *(RQ1, RQ2)*

However, this is not enough for explanations to be useful in practice. **Retrieval over explanations will be learnable to the extent that:**

(2) Explanations are linked to query data points by an easily computable *index* feature *(RQ5)*, and
(3) There are explanations that are sufficiently *relevant* across data points as to be useful for predicting labels for future data *(RQ2, RQ5)*, and
(4) There is a known or identifiable *interpretation* of the explanations by the classifier that yields a useful representation of the explanation *(RQ3, RQ6)*, and
(5) Before training the retrieval model, the classifier reaches some sufficient quality *(RQ7)*, and
(6) Before training the classifier, the initial retrieval model exhibits some sufficient quality *(RQ7)*.

We wish to emphasize a few related results. One of the most intuitive use cases for explanations is to help a model distinguish between strong, generalizable features and weak, spurious features. But **explanations only help break ties between strong and weak features when the model already knows how to interpret the explanations**. When strong and weak features are perfectly correlated, we find that our synthetic explanations do not lead the model to select the causal feature more often than a non-causal one, even when using the optimal retrieval model. We only see that the model can learn to interpret the explanations when the features in question are not perfectly correlated. It seems that, in the paradigm of large language model pretraining, this interpretation function will be meta-learned during pretraining. This behavior is clearly exemplified in GPT-3, which learns from pretraining to infer novel tasks from prompts that precede tasks in zero-shot settings (Brown et al., 2020). Even GPT-2 learns some tasks such as summarization during pretraining, which can be elicited with the right prompt (e.g. "tl;dr") (Radford et al., 2019). As we observe in our experiments, finetuning may allow the model to further identify the correct interpretation, provided that it is identifiable and sufficient training data is available.

It is also important to reiterate that **when using a retrieval model, the information that explanations provide can be inferred from the input alone**. A model need only learn the map between input and hidden information, rather than first using the input for retrieval and then interpreting the retrieved explanation. This is clearly possible in our synthetic task given the relationship between the *index* and (m, n, r, d) values. The same situation will hold true of explanations for real-world tasks. Similar to our synthetic setting, if they can learn to retrieve explanations and then interpret them, they could instead learn to infer the latent information directly from the input. This property of tasks and explanations suggests that conditioning on explanations is a way to structure model computation, biasing it toward desirable functions, and away from difficult to learn functions. That is, as discussed in Sec. 2.3, we see explanations acting as *priors* as well as simply inputs.

So should we collect explanations to assist with solving tasks? At present, the answer is task-specific. In our synthetic task, it is far more helpful to have explanations for 5000 training points than to have $50{,}000$ unexplained points. On benchmark tasks such as e-SNLI, TACRED, and SemEval, we find that explanation retrieval does not yield statistically significant improvements in model accuracy, while using more unexplained data can lead to large improvements. We suggest that the reason for this lies somewhere in the six preconditions for explanation retrieval given above, and it will be useful in future work to develop a diagnostic procedure for further narrowing down the causes of model performance with and without explanations.

More broadly, we see two countervailing trends at work here. The first is that, as language models store more and more knowledge in their parameters, there will be less and less of a need for retrieved explanations to provide hidden information for tasks, though retrieval may still make "accessing" this knowledge easier. In the other direction, we note that as language models become better at interpreting explanations and task descriptions, we will find that for some tasks performance is greatly boosted by having a good task description or set of explanations for example data points.

8. Conclusion

In this paper we present a formal framework for understanding the role of explanations in modeling, and we argue that explanations are most suitably used in a retrieval-based modeling approach, where past explanations are retrieved and used as model inputs for predicting future data points. We experimentally study the preconditions for explanations' usefulness in modeling, and based on results from our synthetic task, we suggest that the model must be able to better infer relevant latent information given the explanation and input, relative to using the input alone. For explanation retrieval to be learnable, we find that (1) explanations should be linked to query data points by an easily computable feature, (2) explanations should be relevant *across* data points, (3) the interpretation of explanations by the classifier should be known or identifiable, and (4) the classifier and retrieval model must both be of some sufficient quality before the other is trained. When we test our method on three existing datasets (e-SNLI, TACRED, and SemEval), we find that explanations do not improve task performance, suggesting that these settings do not meet one of criteria outlined above.

Acknowledgements

We thank Miles Turpin and Ethan Perez for helpful discussion of the topics represented here, as well as Xiang Zhou and Prateek Yadav for feedback on this paper. This work was supported by NSF-CAREER Award 1846185, DARPA Machine-Commonsense (MCS) Grant N66001-19-2-4031, Royster Society 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.

References

Andreas, J., Klein, D., and Levine, S. Learning with latent language. In Walker, M. A., Ji, H., and Stent, A. (eds.), *NAACL-HLT 2018*, 2018. doi: 10.18653/v1/n18-1197. URL https://doi.org/10.18653/v1/n18-1197.

Awasthi, A., Ghosh, S., Goyal, R., and Sarawagi, S. Learning from rules generalizing labeled exemplars. In *ICLR 2020*, 2020. URL https://arxiv.org/pdf/2004.06025.pdf.

Ba, L. J., Swersky, K., Fidler, S., and Salakhutdinov, R. 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*, pp. 4247–4255. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.483. URL https://doi.org/10.1109/ICCV.2015.483.

Baldini Soares, L., FitzGerald, N., Ling, J., and Kwiatkowski, T. Matching the blanks: Distributional similarity for relation learning. In *ACL*, pp. 2895–2905, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1279. URL https://www.aclweb.org/anthology/P19-1279.

Bao, Y., Chang, S., Yu, M., and Barzilay, R. Deriving machine attention from human rationales. In *EMNLP*, pp. 1903–1913, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1216. URL https://www.aclweb.org/anthology/D18-1216.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. A large annotated corpus for learning natural language inference. In *EMNLP 2015*, 2015. URL https://arxiv.org/abs/1508.05326.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *NeurIPS*, 2020. URL https://arxiv.org/abs/2005.14165.

Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. e-snli: Natural language inference with natural language explanations. In *NeurIPS 2018*, 2018. URL https://arxiv.org/pdf/1812.01193.pdf.

Co-Reyes, J. D., Gupta, A., Sanjeev, S., Altieri, N., Andreas, J., DeNero, J., Abbeel, P., and Levine, S. Guiding policies with language via meta-learning. In *ICLR 2019*, 2019. URL https://openreview.net/forum?id=HkgSEnA5KQ.

De Cao, N., Schlichtkrull, M. S., Aziz, W., and Titov, I. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *EMNLP*, pp. 3243–3255, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.262. URL https://www.aclweb.org/anthology/2020.emnlp-main.262.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*, 2017. URL https://arxiv.org/pdf/1702.08608.pdf.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 2020. URL http://proceedings.mlr.press/v119/guu20a.html.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. In *ICLR 2017*, 2017. URL https://openreview.net/pdf?id=rkpACe1lx.

Hancock, B., Varma, P., Wang, S., Bringmann, M., Liang, P., and Ré, C. Training classifiers with natural language explanations. In *ACL*, 2018. URL https://pubmed.ncbi.nlm.nih.gov/31130772/.

Hase, P., Zhang, S., Xie, H., and Bansal, M. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? In *Findings of EMNLP*, 2020. URL https://arxiv.org/abs/2010.04119.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 33–38, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S10-1006.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2017. URL https://arxiv.org/pdf/1702.08734.pdf.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

Kumar, S. and Talukdar, P. Nile : Natural language inference with faithful natural language explanations. In *ACL 2020*, 2020. URL https://arxiv.org/abs/2005.12116.

Lewis, P. S. H., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *NeurIPS*, 2020. URL https://arxiv.org/abs/2005.11401.

Liang, W., Zou, J., and Yu, Z. ALICE: active learning with contrastive natural language explanations. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *EMNLP*, pp. 4380–4391. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.emnlp-main.355/.

Liu, N. F., Lee, T., Jia, R., and Liang, P. Can small and synthetic benchmarks drive modeling innovation? a retrospective study of question answering modeling approaches. *CoRR*, 2021. URL https://arxiv.org/pdf/2102.01065.pdf.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL https://arxiv.org/pdf/1907.11692.pdf.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2017.

Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019. doi: 10.1016/j.artint.2018.07.007. URL https://doi.org/10.1016/j.artint.2018.07.007.

Murty, S., Koh, P. W., and Liang, P. Expbert: Representation engineering with natural language explanations. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *ACL*, pp. 2106–2113. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.190/.

Narang, S., Raffel, C., Lee, K. J., Roberts, A., Fiedel, N., and Malkan, K. WT5?! training text-to-text models to explain their predictions. *ArXiv*, abs/2004.14546, 2020. URL https://arxiv.org/pdf/2004.14546.pdf.

Pruthi, D., Dhingra, B., Soares, L. B., Collins, M., Lipton, Z. C., Neubig, G., and Cohen, W. W. Evaluating explanations: How much do explanations from the teacher aid students? *CoRR*, abs/2012.00893, 2020. URL https://arxiv.org/abs/2012.00893.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. In *OpenAI Technical Report*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Rajani, N. F., McCann, B., Xiong, C., and Socher, R. Explain yourself! leveraging language models for commonsense reasoning. In *ACL 2019*, 2019. URL https://arxiv.org/pdf/1906.02361.pdf.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://www.aclweb.org/anthology/D19-1410.

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model? In *EMNLP*, pp. 5418–5426, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.437. URL https://www.aclweb.org/anthology/2020.emnlp-main.437.

Rupprecht, C., Laina, I., Navab, N., Harger, G. D., and Tombari, F. Guide me: Interacting with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018. URL https://arxiv.org/abs/1803.11544.

Small, K., Wallace, B. C., Brodley, C. E., and Trikalinos, T. A. The constrained weight space svm: learning with ranked features. In *ICML*, pp. 865–872, 2011.

Srivastava, S., Labutov, I., and Mitchell, T. Learning classifiers from declarative language. In *NeurIPS 2017*, 2017. URL http://www.cs.cmu.edu/~shashans/papers/srivastava17-lldworkshop.pdf.

Srivastava, S., Labutov, I., and Mitchell, T. Zero-shot learning of classifiers from natural language quantification. In *ACL 2018*, July 2018. doi: 10.18653/v1/P18-1029. URL https://www.aclweb.org/anthology/P18-1029.

Talmor, A., Tafjord, O., Clark, P., Goldberg, Y., and Berant, J. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *NeurIPS 2020*, 2020. URL https://arxiv.org/abs/2006.06609.

Wang, C., Liang, S., Zhang, Y., Li, X., and Gao, T. Does it make sense? and why? a pilot study for sense making and explanation. In *ACL 2019*, 2019a. URL https://arxiv.org/pdf/1906.00363.pdf.

Wang, Z., Qin, Y., Zhou, W., Yan, J., Ye, Q., Neves, L., Liu, Z., and Ren, X. Learning from explanations with neural execution tree. In *ICLR*, 2019b. URL https://openreview.net/pdf?id=rJlUt0EYwS.

Weller, O., Lourie, N., Gardner, M., and Peters, M. Learning from task descriptions. In *EMNLP*, pp. 1361–1375, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.105. URL https://www.aclweb.org/anthology/2020.emnlp-main.105.

Wiegreffe, S., Marasovic, A., and Smith, N. A. Measuring association between labels and free-text rationales. *CoRR*, abs/2010.12762, 2020. URL https://arxiv.org/abs/2010.12762.

Zaidan, O., Eisner, J., and Piatko, C. 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*, pp. 260–267, Rochester, New York, April 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N07-1033.

Zhang, Y., Marshall, I., and Wallace, B. C. Rationale-Augmented Convolutional Neural Networks for Text Classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 795–804, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1076. URL https://www.aclweb.org/anthology/D16-1076.

Zhang, Y., Zhong, V., Chen, D., Angeli, G., and Manning, C. D. Position-aware attention and supervised data improve slot filling. In *EMNLP*, pp. 35–45, 2017. URL https://nlp.stanford.edu/pubs/zhang2017tacred.pdf.

Zhou, W., Hu, J., Zhang, H., Liang, X., Sun, M., Xiong, C., and Tang, J. Towards interpretable natural language understanding with explanations as latent variables. In *NeurIPS*, 2020. URL https://arxiv.org/pdf/2011.05268.pdf.

e-SNLI Example 1	
x :	*Premise:* After playing with her other toys, the baby decides that the guitar seems fun to play with as well. *Hypothesis:* A blonde baby.
y :	Neutral
e :	Not all babies are blonde.
e-SNLI Example 2	
x :	*Premise:* A girl wearing a pink and black shirt and jeans fixes her hair before walking up the stairs. *Hypothesis:* A girl has blonde hair.
y :	Neutral
e :	Not all girls have blonde hair.
SemEval Example 1	
x :	The SUBJ originates from an OBJ which transcends the speaker.
y :	Entity-Origin
e :	The phrase ”originates from an” occurs between SUBJ and OBJ and there are no more than four words between SUBJ and OBJ and OBJ follows SUBJ.
SemEval Example 2	
x :	With one exception, the SUBJ emerged from the OBJ during hours of darkness.
y :	Entity-Origin
e :	The phrase “emerged from the” occurs between SUBJ and OBJ and there are no more than four words between SUBJ and OBJ and SUBJ precedes OBJ.
TACRED Example 1	
x :	SUBJ’s husband OBJ died in 1995.
y :	Person-Spouse
e :	Between SUBJ and OBJ the phrase “’s husband” occurs and there are no more than five words between SUBJ and OBJ.
TACRED Example 2	
x :	SUBJ is married to OBJ and is the father of three sons.
y :	Person-Spouse
e :	There are no more than four words between SUBJ and OBJ and the phrase “is married to” appears between SUBJ and OBJ.

Table 4. Additional example data points from three existing datasets.

A. Training Details

A.1. Data Preprocessing

No preprocessing is applied to the synthetic data. For the three existing datasets, we use maximum sequence lengths as follows: For e-SNLI, we use a maximum sequence length of 120 tokens, with maximum lengths of 90 for x and 60 for each explanation. For TACRED and SemEval, we use a max of 160 for the entire input, with a max of 80 for x and 60 for e. We remove one explanation from the set of explained data points for TACRED after finding that it is given for a data point in the dev set.

We give additional examples of data points from each dataset in Table 4.

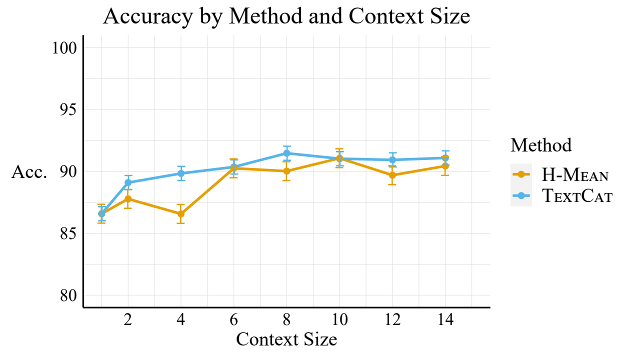


Figure 11. (*Training Hyperparameters and Analysis.*) Learned retrieval accuracy by C, with $k = 1$ and optimal retrieval, for our two conditioning mechanisms.

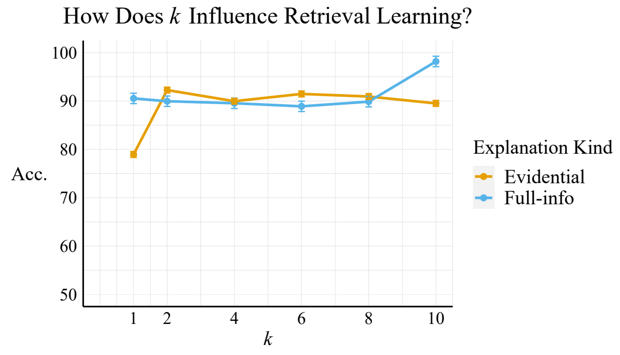


Figure 12. (*Training Hyperparameters and Analysis.*) Learned retrieval accuracy by k, with $C = 1$, for *full-info* and *evidential* explanations.

A.2. Training Hyperparameters and Analysis

For optimization, we use AdamW with a learning rate of $1\text{e}{-5}$ 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.

Condition	Model	Acc.
e-SNLI		
n=10,987	TEXTCAT-E	87.17 (0.66)
	TEXTCAT-YXE	87.11 (0.66)
SemEval		
n=7016	TEXTCAT-YE	75.25 (2.99)
	TEXTCAT-YXE	75.75 (2.97)
TACRED		
n=68,124	TEXTCAT-YE	87.49 (0.43)
	TEXTCAT-YXE	87.31 (0.43)

Table 5. Ablation across the retrieved variables: the suffix to TEXTCAT indicates which retrieved variables are included in the model input. We do not find that including x improves model performance, so we use only e or (y, e), depending on the task.

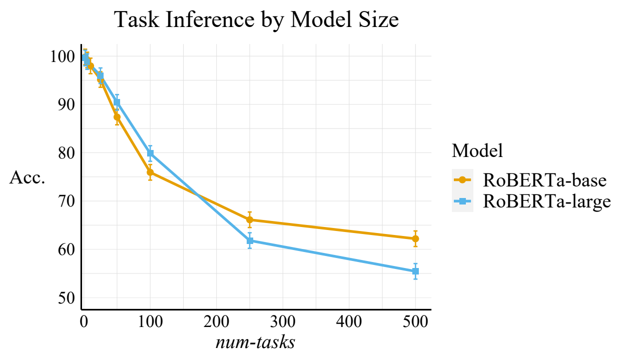


Figure 13. (**RQ1**) Synthetic task accuracy as a function of *num-tasks*, by model size.

We measure the relationship between the context size C and performance on evidential explanations, using the optimal retrieval model and comparing between conditioning mechanisms. The results are shown in Fig. 11. We see that, with each method, a larger value of C is preferable up to around 8 or so, after which performance plateaus.

Regarding the value of k, we see in Fig. 12 that training performance can be sensitive to the chosen value for this hyperparameter. It appears that one should try to select as high a value of k as possible, all else equal. Though since this parameter increases the number of forward passes during training by a factor of k, there is a trade-off between the available compute budget and the value of k in practice.

A.3. Model Selection in Experiments

In general, within a single training run, we select the model that achieves the best dev set accuracy as measured at the end of each training epoch.

With our synthetic task, we observe some training instability in a few conditions, particularly in those where we are degrading the training model (RQ7). On such occasions, training fails after a few epochs and model accuracy trends toward 50% (random performance). These these are easily noticeable, we rerun these experiments with a different seed in order to report results from a stable run, and typically we find that stable training dynamics can be obtained from just one other seed.

For the existing datasets, we run three model seeds for the baseline and each of the hyperparameter conditions, except for when using at least 50,000 training data points, where we run only one seed. We also use only one seed when ablating across which retrieved variables to include in the model input (i.e. whether to include the retrieved x in the model input). To select one model from three seeds for a given condition, we pick based on the highest dev performance.

The results for ablating across the retrieved variables to include as model inputs are shown in Table 5. Note that we test the effect of adding x to e for e-SNLI and the effect of adding x to (y, e) for relation extraction tasks, since y is easily inferred from e for e-SNLI (Hase et al., 2020). In these experiments, we roughly control for the sequence length, meaning that for relation extraction tasks, we use $C = 1$ when x is present and $C = 2$ when it is not, while for NLI we use $C = 5$ without x and $C = 2$ with x. These experiments all use $k = 4$. We do not find any statistically significant differences in dev set accuracy across any of the conditions. Hence, we proceed with using (y, e) for relation extraction tasks and e for NLI.

When tuning over C and k with the existing datasets, we use the following values for each condition: For relation extraction tasks, we tune over values in $\{(C{=}2, k{=}4),(C{=}1, k{=}8)\}$, and for NLI we tune over $\{(C{=}4, k{=}4),(C{=}2, k{=}8)\}$ when $n < 50000$. We tune separately for each training set size configuration. We select the hyperparameters to use based off of the best dev set accuracy achieved from three seeds in each condition. We report results from a single run of $(C{=}2, k{=}4)$ for e-SNLI with the full training data.

B. Experimental Details And Additional Results

In this section, we describe experimental details and hyperparameters for particular experiments, organized by research question, as well as additional results accompanying some research questions. Lastly, we discuss hypothesis testing procedures. Unless otherwise stated, additional experiments below use the default synthetic task parameters, given in the Synthetic Task section in the main paper.

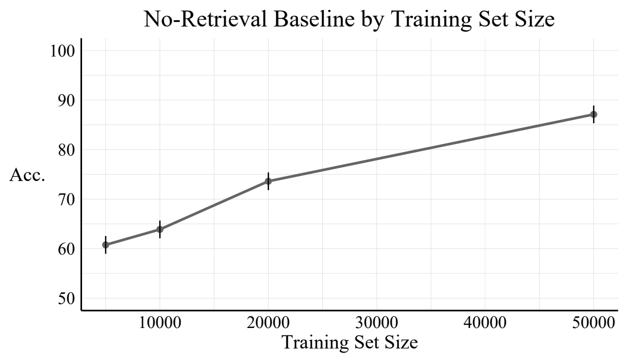


Figure 14. (**RQ1**) Synthetic task accuracy as a function of the training set size, without explanation retrieval.

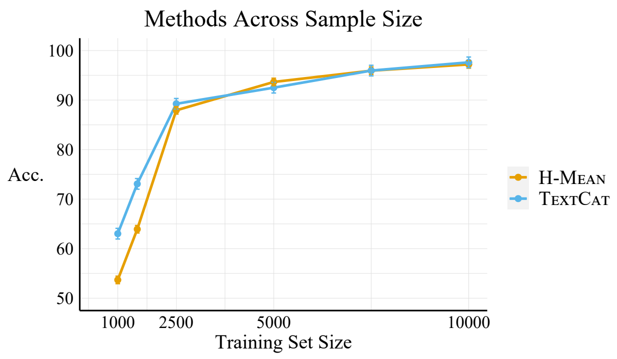


Figure 15. (**RQ4**) Synthetic task accuracy across training set sizes and method, using optimal retrieval.

B.1. RQ1: When can models solve our synthetic problem by inferring each sequence's task, and when must they be given the task information?

We show results for additional model choices and sizes of the training datasets here. In Fig. 13, we see that RoBERTa-large outperforms RoBERTa-base only when the number of tasks in the training data is relatively small. After *num-tasks*$\geq$100, RoBERTa-base is the better choice. Here, we train models for 40 epochs with a LR of $1\text{e}{-5}$, though for *num-tasks*$\in \{250, 500\}$ with RoBERTa-large, we have to train for 60 epochs with a LR of $1\text{e}{-6}$ in order for training to converge.

In Fig. 14, we see that performance scales well with the available training data for our synthetic task. RoBERTa-base reaches an accuracy of 87.11 with 50,000 training points.

B.2. RQ2: Can retrieval of past explanations enable a model to solve our task?

In these experiments, we always freeze the retriever for the first two epochs of training and train for a total of 20 epochs.

B.3. RQ3: Can models aggregate information across explanations for better prediction?

Here, we freeze the retriever for the first five epochs of training and train for a total of 25 epochs.

B.4. RQ4: What is the best way to compute explanation representations for prediction?

In Fig. 15, we see that TextCat outperforms H-Mean at smaller training set sizes. TextCat achieves higher accuracy than H-Mean by 9.3 points for $n = 1000$ and 9.2 points for $n = 1500$, though the gap shrinks to 1.3 points at $n = 2500$ and the methods perform equally well after $n = 5000$. In these experiments we use the optimal retrieval model.

B.5. RQ5: What makes an explanation relevant across data points? What enables a retrieval model to find relevant explanations for a new data point?

In these experiments, training hyperparameters match those in RQ2, except for details regarding the n_{task} and *smoothness* conditions. In order to achieve a smooth function from *index* to (m, n), we first order the domain and codomain and then match them up one-to-one. To order (m, n) tuples, we sort by m first and then n. The result is that when two explanations have similar *index* values, their m values are very likely to be close together, and their n values will probably be close together. Note that in this experiment, we sample (m, n) not from $unif([1, 100]^2)$, but rather we draw the valid (m, n) tuples in increasing order starting from the first valid tuple, $(1, 2)$. We use the same (m, n) sampling scheme for the baselines and the *non-smooth* condition. The only difference in the *non-smooth* conditions is the lack of ordering in the domain and codomain before they are matched up. This allows for more precise inference as to the task parameters when retrieving an explanation with a similar *index* to a data point at hand.

B.6. RQ6: Can explanations help models learn to use strong features rather than weak ones?

We give additional results with the strong-weak feature correlation varied between 0 and 1 in Fig. 16, using the training hyperparameters for RQ2. Using the *full-info* explanation with optimal retrieval, we see the model continues to perform well as long as the features are not perfectly correlated. Interestingly, the no-retrieval condition's performance rises as the correlation increases, though it never matches the retrieval condition's performance. Since the performance of optimal retrieval is above the baseline but not greater than 75% when the features are perfectly correlated, the explanations are not helping decide whether to use the strong or weak feature, but they are helping these features be used in the first place (see the footnote in the results for RQ5 in the main body).

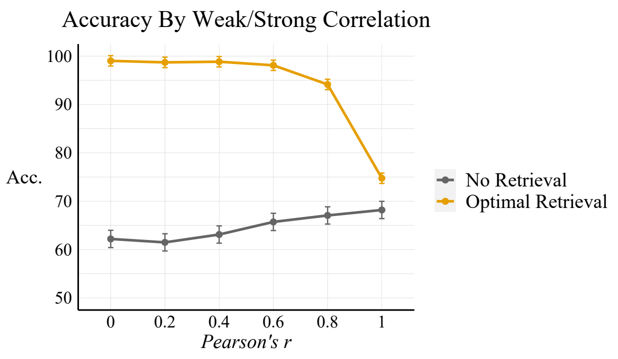


Figure 16. (**RQ6**) Synthetic task accuracy across correlation levels between the strong and weak feature, with and without optimal retrieval.

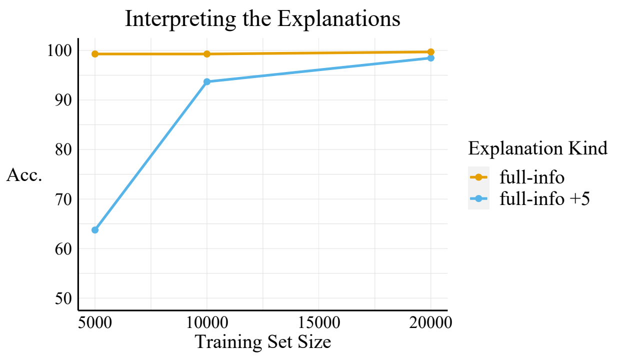


Figure 17. (**RQ6**) Even a simple function on explanations can render them difficult for the model to interpret, though the correct interpretation is identified with more data.

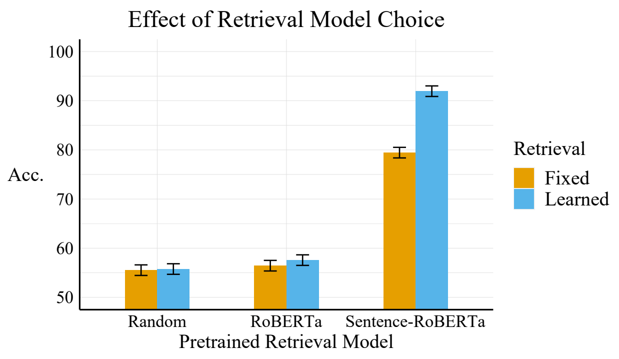


Figure 18. (**RQ7**) Model performance by choice of retriever. Using a pretrained Sentence-BERT model is vital to the success of learning a retrieval model in our synthetic task.

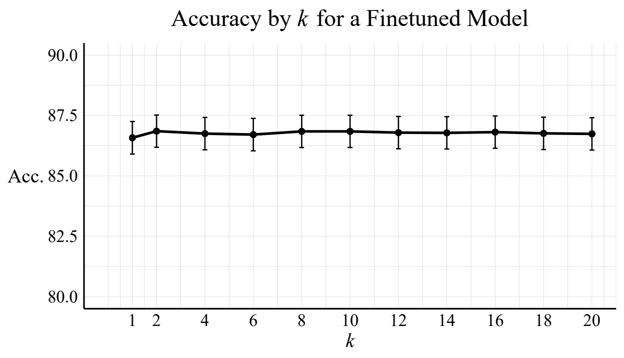


Figure 19. (**RQ8**) Dev set accuracy across k for the retrieval model on e-SNLI using 10000 training points.

It may even be surprising that the *full-info* explanations are useful when the strong-weak correlation is 0, since the Causal Integer explanations are not. In Fig. 17, we see that, while the correlation is 0, some explanations may be hard to interpret when a small amount of training data is available, but as more data is available the correct interpretation is identified. Here, we simply add 5 to each integer in the *full-info* explanations. Using optimal retrieval, models struggle to correctly interpret these explanations at a low sample size, but with more data the correct interpretation is identified.

B.7. RQ7: How does the co-dependence between classifier and retrieval model influence the viability of joint training?

Hyperparameters in experiments for this RQ match those for RQ3. In Fig. 18, we show the effect of the retrieval model choice on the viability of learning retrieval. As in the main body, we also use evidential explanations with $\epsilon = 2$. We find that it is necessary to use a pretrained Sentence-RoBERTa model. Simply using a pretrained RoBERTa-base model will not suffice for learning retrieval with our synthetic task. Surprisingly, this condition cannot outperform even a randomly initialized model with an identical architecture. This could be due to the fact that we use the mean token pooling and cosine similarity that the Sentence-BERT models were trained with.

B.8. RQ8: Does retrieval of explanations improve model performance on existing datasets?

We train for: 5 epochs when using the full e-SNLI training set; 20 epochs when using $n \leq 10000$ for any dataset; and 10 epochs for other larger values of n.

In Fig. 19, we show the result of varying the value of k used to calculate dev set accuracy for the retrieval model in the e-SNLI with $n = 10000$ condition. We see no meaningful changes in dev set accuracy across values of k from 1 to 20, showing that increasing k at test time is not a reliable way to improve retrieval model accuracy in this setting.

Lastly, we observe that the ELV-M condition from Zhou et al. (2020), which is H-MEAN with fixed retrieval and (C=10, k=1), does not outperform baselines on TACRED and SemEval. The approach obtains 87.99% on TACRED, where our baseline is 88.29%, and 76.46% on SemEval, where the baseline is 76.94%. Besides using RoBERTa models instead of BERT, one change we make from the implementation in Zhou et al. (2020) is to disallow for data

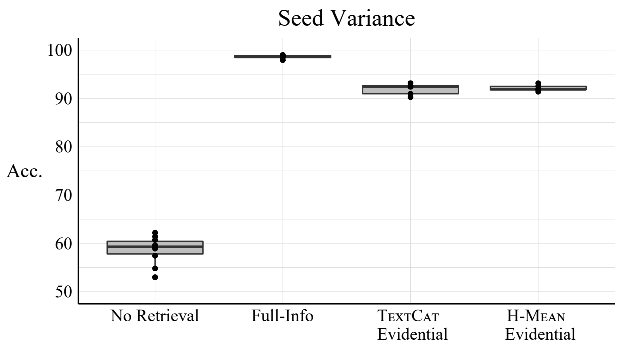


Figure 20. (Seed variance for some representative experimental conditions.

points' own explanations to be conditioned on when predicting their labels, although this is not relevant for predicting test points in either dataset.

B.9. Confidence Intervals and Hypothesis Testing

We compute confidence intervals for our synthetic data tasks to represent *seed variance* around some mean seed performance, while confidence intervals and associated hypothesis tests for existing datasets represent *sample variance.* With synthetic data 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. Then we run 5 runs with learned retrieval using (1) TextCat with *full-info* explanations, (2) TextCat with *evidential* explanations, and (3) H-Mean with *evidential* explanations. The results of these runs are shown in Fig. 20. We then assume that seed variance is invariant across experimental factors not related to the choice of conditioning method or explanation and assign 95% confidence intervals across experimental conditions based on these four representative conditions. We prioritize assignments based on the explanation kind (*full-info* vs. *evidential* or *recomposable*), then by conditioning mechanism, when for instance some conditions use combinations of methods and explanation kinds not represented in these conditions. We assume these invariances in order to efficiently calculate seed variance. Running 5 seeds per retrieval condition and 10 per non-retrieval would increase the number of synthetic data experiments in this paper from 172 to 860. In synthetic data experiments, we comment on effects far larger than the confidence intervals and do not conduct hypothesis tests.

The confidence intervals shown for model accuracies on existing datasets are 95% confidence intervals on the underlying binomial probability. The hypothesis tests conducted for RQ8 are two-sided difference in binomial means tests.

C. 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 RQ5, *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 RQ5.

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 $\{(\textit{index}, m, n, r, d)_{\textit{index}})\}$ 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* $\in \{\textit{index}_t\}_{\tau=1}^{\textit{num-tasks}}$:
   (a) Sample a vector of length n_{task}, balanced between 1s and 2s, that gives the values of $\{\textit{indicator}_p\}_{p=1}^{P}$ 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 $(m, n, r, d)_{\textit{index}}$ 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 $unif(1, 100)$.