Breakpoint Transformers for Modeling and Tracking Intermediate Beliefs

Kyle Richardson[2†] **Ronen Tamari**[1†*] **Oren Sultan**[1]
Reut Tsarfaty[2,3] **Dafna Shahaf**[1] **Ashish Sabharwal**[2]

[1]The Hebrew University of Jerusalem [2]Allen Institute for AI [3]Bar-Ilan University
{ronent,orens,dshahaf}@cs.huji.ac.il, {kyler,reutt,ashishs}@allenai.org

Abstract

Can we teach natural language understanding models to track their beliefs through intermediate points in text? We propose a representation learning framework called breakpoint modeling that allows for learning of this type. Given any text encoder and data marked with intermediate states (*breakpoints*) along with corresponding textual queries viewed as true/false propositions (i.e., the candidate *beliefs* of a model, consisting of information changing through time) our approach trains models in an efficient and end-to-end fashion to build intermediate representations that facilitate teaching and direct querying of beliefs at arbitrary points alongside solving other end tasks. To show the benefit of our approach, we experiment with a diverse set of NLU tasks including relational reasoning on CLUTRR and narrative understanding on bAbI. Using novel belief prediction tasks for both tasks, we show the benefit of our main *breakpoint transformer*, based on T5, over conventional representation learning approaches in terms of processing efficiency, prediction accuracy and prediction consistency, all with minimal to no effect on corresponding QA end-tasks. To show the feasibility of incorporating our belief tracker into more complex reasoning pipelines, we also obtain SOTA performance on the three-tiered reasoning challenge for the TRIP benchmark (around 23-32% absolute improvement on Tasks 2-3).[1]

1 Introduction

Despite considerable progress made recently in natural language understanding (NLU), driven largely by advances in language model pre-training (Devlin et al., 2019; Raffel et al., 2020) and the development of large-scale NLU benchmarks (Wang et al., 2018), understanding the behavior of models remains a formidable and highly consequential challenge for model safety. Such a challenge is particularly acute in tasks such as narrative understanding, where one must piece together many individual (possibly implicit) facts through time in order to solve problems. For example, in the story in Figure 1, answering the question *Where is the apple?* requires knowing how to track objects through time (e.g., knowing the location of the *John* and *Mary* and their interaction) and how to compartmentalize other types of knowledge across the story. In such a setting, where models are trained to narrowly answer questions, a natural question arises: *do models acquire the kind of requisite background knowledge and world tracking abilities, and ultimately learn representations that give rise to correct* beliefs[2] *about intermediate states?*

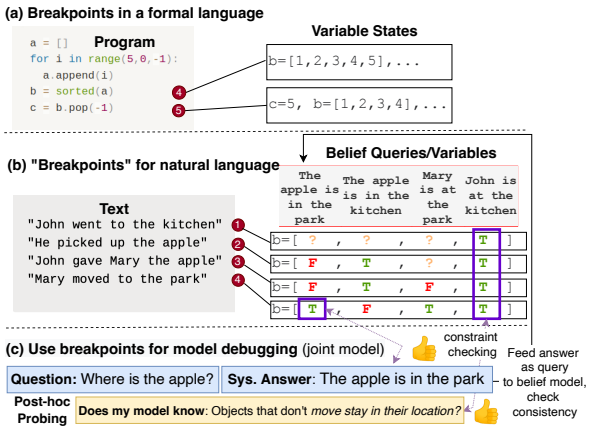


Figure 1: Deep narrative understanding in natural language (bottom) involves the ability to answer **queries** about arbitrary intermediate points in a given story. We liken this task to *breakpoints* in programming (top), or reporting the state of a program at different stages of execution, facilitating human inspection of model beliefs and consistency with end-task behavior (bottom).

A chief difficulty in answering such questions is

*Work begun during an internship at the Allen Institute.
†Equal contribution.
[1]Project code available at https://github.com/allenai/situation_modeling.

[2] Similar in spirit to Kassner et al. (2021), we define a *belief* as an attribution of a truth value to a proposition relative to a context or *partial information state* (Landman, 2012). E.g., a belief that *John is in the kitchen* is `true` in the context immediately following the event *John went to the kitchen*.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 9703–9719
December 7-11, 2022

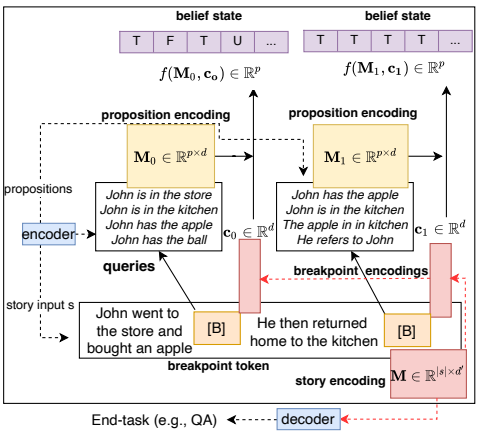


Figure 2: A high-level view of our modeling approach. For a given story and a set of textual **queries** corresponding to intermediate points in the story (**breakpoints**), truth assignments are assigned to queries to form **belief states** based on a projection over encodings of breakpoints and individual **proposition encoding**s using a single task-specific **encoder**.

that directly inspecting the propositional attitudes of our current models remains a formidable challenge due to the latent nature of their knowledge. Such a complication also makes it unclear what the right interface should be for eliciting beliefs in the first place (e.g., how can we determine if a model believes a proposition *John is in the kitchen* at an arbitrary point in text?). In addition, for tasks such as QA, story contexts and questions are usually encoded jointly (often with full attention over context and query), which makes it difficult to tease apart a model's understanding of a story independent of each question. Entangled story and question representations can be inefficient when scaling to a large space of questions, particularly for novel combinations of questions and stories (Tamari et al., 2022). Such entangled representations also allow models to exploit spurious patterns in questions that inflate performance (Kaushik and Lipton, 2018) and hinder interpretability.

We present a model-agnostic representation learning framework called *breakpoint modeling* that facilitates teaching models to have propositional beliefs at arbitrary points in stories (or *breakpoints*) using ordinary textual queries as our interface language. Our general modeling approach is illustrated in Figure 2. Given any task-specific encoder and data marked with the intermediate state of interest (or *breakpoints*, denoted throughout as **[B]**) along with a set of textual queries (i.e., the candidate *beliefs* provided in training as auxiliary intermediate supervision), models are trained in an end-to-end fashion to learn intermediate task-specific representations (pooled from single encodings of stories) that jointly facilitate making correct and consistent belief predictions efficiently across a large space of queries. Making an analogy with *breakpoints* in programming (see top of Figure 1), we aim to *simulate* stopping execution at intermediate points during a story to inspect the model's belief state (e.g., checking that a model's answers for QA are consistent with their beliefs and satisfy certain high-level constraints), as well as teach the model to have certain beliefs learned through intermediate supervision at training time.

Using a state-of-the-art pretrained model, T5 (Raffel et al., 2020), we develop and investigate a *breakpoint transformer* to do belief prediction on three categories of tasks: *narrative understanding* on bAbI (Weston et al., 2016; Tamari et al., 2022), *relational reasoning* on CLUTRR (Sinha et al., 2019) and *physical commonsense reasoning* over human authored stories on TRIP (Storks et al., 2021). In the former two cases, we focus on training and evaluating models on a novel belief prediction task. We report improvements over a conventional transformer-based representation learning approach (Reimers and Gurevych, 2019) both in terms of prediction accuracy (4% to 8% absolute improvement on CLUTRR dev) and belief consistency, all with significantly improved processing efficiency (i.e., minimal forward calls to the full transformer) and minimal effect on end-task performance when jointly trained with QA. In the latter case for TRIP, we show how to integrate our modeling approach into a more complex transformer pipeline and report state-of-the-art results on the three-tiered reasoning task (with 23-32% absolute improvement on two component tasks) over existing task-specific architectures.

Taken together, our results show the viability of building an end-to-end trainable belief tracking mechanism and integrating it within existing transformer-based reasoning systems. To our knowledge, our work is among the first to look at at general-purpose sentence representation learning for intermediate states in text as a way to facilitate complex situation reasoning.

2 Related Work

Our work brings together two recent areas that aim to understand model behavior (broadly *model prob-*

Task	Example Stories	Breakpoint Propositions
Relational Reasoning (CLUTRR)	John is the brother of Susan **[B]**$_1$ Susan's mother is Janice **[B]**$_2$, ...	$\mathbf{P}_1$: { 'Susan is the sister of John' **true**, 'Susan is the sister-in-law of Janice' **false**,'Janice is the mother of John' unk } $\mathbf{P}_2$: { 'Janice is the mother of John' **true**, 'John is the father of Janice' **false**, ...}
Story Understanding (bAbI)	John moved to the kitchen **[B]**$_1$ He picked up an apple **[B]**$_2$ John then gave the apple to Mary **[B]**$_3$ ...	$\mathbf{P}_1$:{ 'John has the apple' **false**, 'John is in the kitchen' **true**,...} $\mathbf{P}_2$: { 'John has the apple' **true**,'John is in the kitchen'...}... $\mathbf{P}_3$: { 'John has the apple' **false**, 'Mary has the apple' **true** }
Commonsense (TRIP)	Tom dropped his radio ...carpet. **[B]**$_1$ The radio broke .. **[B]**$_2$ Tom turned on the radio ... **[B]**$_3$ ...	$\mathbf{P}_2$: { 'radio is in pieces' **true**, 'radio is powered' **false**}, ... $\mathbf{P}_3$: { 'radio was powered' **true** }

Figure 3: Three tasks rendered as **stories** with special breakpoint tokens **[B]**$_j$ (for convenience, marked with an index j). Each intermediate breakpoint is aligned to a set of propositions $\mathbf{P}_j$ marked with truth conditions (i.e., **true**, **false**, unknown) corresponding to the truth value of each proposition at that breakpoint.

ing): probing of the type that includes finding neural correlates of high-level behavioral phenomena, modular structure in networks (Tenney et al., 2019; Hewitt and Manning, 2019) on the one hand, as well as diagnostic testing, which aims to understand model competence through controlled input-output testing (Lake and Baroni, 2018; Richardson et al., 2020), or post-hoc consistency analysis (Kassner et al., 2021). Our work is more closely related to Li et al. (2021), who show that partial world state information can be decoded from NLMs even without explicit supervision. In that work, state information is roughly localized to entity mentions, but varies across different datasets. Differently from such probing work, our breakpoint models are trained in a supervised manner to localize particular propositional information at particular locations (similar to Geiger et al. (2021)).

Our breakpoint model closely relates to *late-interaction encoder* architectures that tease apart the encoding of problems and solutions. This includes the sentence transformer from Reimers and Gurevych (2019), which we compare against in our experiments, as well as *read once transformers* (Lin et al., 2021), colBERT (Khattab and Zaharia, 2020) and others. Given that the types of narrative tasks we focus on require modeling many intermediate points, we follow this work in putting an emphasis on representation and encoding efficiency. In contrast to this, and other related work on sentence representation learning (Gao et al., 2021; Ni et al., 2022), we uniquely focus on learning representations of intermediate states in text for complex situational reasoning.

We are also inspired by the situation modeling literature in cognitive science (Golden and Rumelhart, 1993; Frank et al., 2003; Venhuizen et al., 2019), and proposals for their integration with NLP research (Tamari et al., 2020). These works also studied neural models of narrative comprehension in carefully controlled micro-worlds, but typically focused on relatively short sentence-level inputs.

Our work also relates to efforts on building interpretable models by making the underlying reasoning processes transparent, either through explicit decomposition (Andreas et al., 2016; Khot et al., 2021; Bostrom et al., 2022) or generation of rationales (Camburu et al., 2018; Wiegreffe and Marasovic, 2021) and other reasoning structures (Tafjord et al., 2021; Dalvi et al., 2021; Gontier et al., 2020). In contrast, we focus on belief representations that are ultimately *faithful* (Jacovi and Goldberg, 2020) to end-tasks by training knowledge directly into a model's task-specific representations.

3 Breakpoint Modeling

The goal of breakpoint modeling is to capture the intermediate states and beliefs of models at arbitrary positions in text. Our models take *stories* as inputs, or pieces of text containing one or more intermediate positions (*breakpoints*), as well as sets of text *propositions* that align to certain intermediate points (see Figure 3). Such propositions play the role of auxiliary supervision if provided at training time or as queries to the model for performing probing; when coupled with predictions they constitute the *beliefs* of the model.

While breakpoint models can technically take different forms, their basic function is to assign encodings to intermediate states in text and their corresponding propositions (§ 3.1) and to make predictions about the truth/falsity of each proposition (§ 3.2). Learning (§ 3.3) reduces to the problem of teaching a model to have a correct and consistent set of beliefs for each target task given a set of representative intermediate propositions and beliefs provided at training time (§ 3.4).

3.1 Breakpoint and Proposition Encoding

As illustrated in Figure 3, **stories** are texts consisting of n tokens within which there can exist $m \geq 1$ arbitrarily selected intermedi-

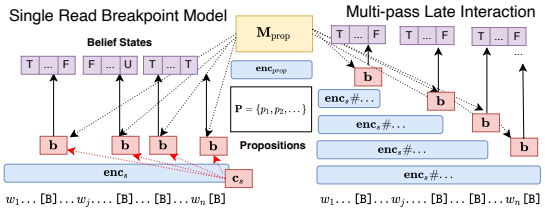


Figure 4: What is the best way to model intermediate states and beliefs with existing encoder models? An illustration of two late-interaction architectures we investigate (**Single Read** model, our main model described in § 3 and a **Multi-pass Late Interaction** model)

ate points or **breakpoints**. For convenience, we will render a story s in the following way: $s := w_{1,b_1} \ldots w_{,b_1}$ **[B]** $\ldots w_{,b_j} \ldots$ **[B]** $\ldots w_{n,b_m}$ **[B]** where **[B]** is a special token used to explicitly mark position of each breakpoint b_j. Intuitively, a breakpoint token represents all of the information in the story relevant to building an accurate belief state at the corresponding (intermediate) point in the text. Associated with each b_j is a set of text **propositions** $\mathbf{P}_j = \{p_1, p_2, ..., p_t\}$. Truth assignments to these text propositions constitute the candidate beliefs at breakpoint b_j (in the sense of Footnote 2).

At the core of any breakpoint model are two encoders, $\mathbf{enc}_{\text{story}}, \mathbf{enc}_{\text{prop}}$, that are used to generate a representation or embedding for each breakpoint in the story and each proposition, respectively. Representations of breakpoints $\mathbf{b} \in \mathbb{R}^d$ are pooled from a single encoding of an input story s: $\mathbf{c}_s \leftarrow \mathbf{enc}_{\text{story}}(s) \in \mathbb{R}^{|s| \times d}$ and representations for propositions $\mathbf{c}_{\text{prop}} \in \mathbb{R}^d$ are obtained in a similar fashion using $\mathbf{enc}_{\text{prop}}$. While the choice of the encoder and the details of how pooling is done can vary (see details in §5.1), in all of our models breakpoint representations $\mathbf{b}$ are obtained by taking projections of the hidden states of the **[B]** tokens from $\mathbf{c}_s$. We also investigate models that assume a *siamese* architecture (Reimers and Gurevych, 2019) where $\mathbf{enc}_{\text{story}}$ and $\mathbf{enc}_{\text{p}}$ are the same encoder.

An important property of breakpoint models is that all breakpoints representations $\mathbf{b}_j$ are obtained from a *single read* encoding of each target story. We later compare this against a much less efficient approach that requires multiple forward passes through the story to obtain intermediate encodings (i.e., the **multi-pass** approach shown in Figure 4). Our model therefore stays within the spirit of a *late-interaction* architecture (Khattab and Zaharia, 2020) by using separate encodings of breakpoints and propositions, which allows us to scale to large sets of propositional queries.

3.2 Proposition Scoring and Semantics

Given a breakpoint encoding $\mathbf{b}$ and an aligned proposition encoding $\mathbf{c}_{\text{prop}}$, a **proposition scorer** makes a prediction about a proposition at that breakpoint. As mentioned, our aim is to predict the truth value of a proposition at an intermediate state, which we take to be the model's *belief* in that proposition. Our scorer takes the form of a classifier that maps a breakpoint encoding and proposition encoding to the discrete space $\{\mathbf{true}, \mathbf{false}, \mathbf{unknown}\}$, following Li et al. (2021) and the annotation scheme from NLI (Dagan et al., 2005; Bowman et al., 2015).

To make clear that the interpretation of each proposition is tied to a specific breakpoint, we will use the symbolic notation from Li et al. (2019) and introduce three binary *logical predicates* $\mathbf{E}$, $\mathbf{C}$, and $\mathbf{U}$. For each b_j and $p \in \mathbf{P}_j$, these predicates capture whether p is **entailed** by, is **contradicted** by, or has an **unknown** relation to the information in the text at breakpoint b_j, respectively. For instance, $\mathbf{E}(b_j, p)$ is true if the text proposition p is entailed by the story at breakpoint b_j.

3.3 Learning

Suppose we have a dataset D consisting of n stories $\{s^{(i)}\}_{i=1}^{n}$ along with the following additional information. For each story $s^{(i)}$, we have m breakpoints $B^{(i)}$.[3] For each such breakpoint b_j, we have t labeled text propositions[4] $\mathbf{P}_j^{(i)}$, where each proposition $p_k \in \mathbf{P}_j^{(i)}$ is labeled with $y_{j,k}^{(i)} \in \{\mathbf{true}, \mathbf{false}, \mathbf{unknown}\}$ indicating p_k's truth value at breakpoint b_j. Using the above predicate logic notation, we can equivalently think of having, for each $p_k \in \mathbf{P}_j^{(i)}$, exactly one predicate $Y_{j,k}^{(i)} \in \{\mathbf{E}, \mathbf{C}, \mathbf{U}\}$ annotated in D, with the semantics that $Y_{j,k}^{(i)}(b_j, p_k)$ is True (and the other two predicates for b_j and p_k are False).

The goal here is to learn a model that assigns truth values to all text propositions across all breakpoints—equivalently, truth values for all three logical predicates—in a way that maximally aligns with D. Semantically, this can be expressed as

[3] In general, m depends on i. However, for simplicity of exposition, we use m here instead of $m^{(i)}$.

[4] Again, t in general depends on both i and j, but we use t instead of $t_j^{(i)}$ here for simplicity.

satisfying the logical formula (Li et al., 2019):

$$\bigwedge_{s^{(i)} \in D} \bigwedge_{b_j \in B^{(i)}} \bigwedge_{p_k \in \mathbf{P}_j^{(i)}} Y_{j,k}^{(i)}(b_j, p_k) \qquad (1)$$

with the added constraint that for each story $s^{(i)}$ and all j, k, exactly one of $\mathbf{E}(b_j, p_k), \mathbf{C}(b_j, p_k)$, and $\mathbf{U}(b_j, p_k)$ is True.

Using $\Pr[y_{j,k}^{(i)}]$ to denote the model's probability corresponding to the predicate $\mathbf{Y}_{j,k}^{(i)}(b_j, p_k)$, this formula can be translated into the following loss using the *product* translation from Li et al. (2019):

$$\mathcal{L}_{\text{prop}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} -\log \Pr[y_{j,k}^{(i)}] \qquad (2)$$

which yields the common cross-entropy loss that we use in our experiments.

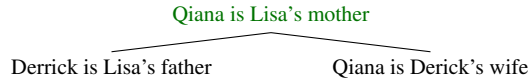
3.4 Proposition Sampling

Propositions in breakpoint models have a dual role: when given at training time, they provide intermediate supervision for training models across different situation states. When given at inference time they allow for post-hoc probing of a model's beliefs. As shown in the Figure 3, propositions, in virtue of being ordinary text, can express many different types of information and thus provide an unbounded source of *semantic supervision* (Hanjie et al., 2022), e.g., for expressing *fluents*, or conditions that change through time in a story (e.g., *John is in the kitchen*, or event pre/post-conditions (e.g., *The radio was powered* via English tense).

For training models to have beliefs, a necessary first step is to devise a **sampling policy** for generating these intermediate annotations. While such a strategy needs to be tailored to each target task, we experiment with a combination of extracting propositions from existing task annotations (Figure 5) and generating propositions based on a set of **domain constraints** using the semantics of each target domain (details in the next section).

4 Proposition Prediction Tasks

We focus on three categories of tasks: text-based **relational reasoning**, **story understanding** and **commonsense reasoning**, each considered in turn. In the former two cases, we devise new proposition and belief prediction tasks that involve training on intermediate belief state annotations. We also include out-of-distribution (o.o.d) generalization tests beyond standard i.i.d (independent and identically distributed) evaluation. In the latter case, we recast an existing task in terms of breakpoint models to show the versatility of our approach in a more complex multi-task setting.



Story	Lisa is Jerry's granddaughter **[B]** Derrick is Lisa's father **[B]** Qiana is Derrick's wife **[B]**
Atomic Belief Annotations: The basic facts that should be predicted	**E**(1,'Jerry is the grandfather of Lisa') **E**(1,'Derick is the father of Lisa'), **C**(1,'Lisa ..father of Derick'), **E**(3,'Qiana..wife of Derick'), **E**(3,Qiana is the mother of Lisa)
Knowledge: Constraints to satisfy	`(Implies(And` **E**(2,'Derick...father of Lisa') **E**(3,'Qiana is the wife of Derick')) **E**(3,'Qiana is the mother of Lisa'))

Figure 5: How are intermediate propositions collected? An illustration of constructing intermediate propositions from CLUTRR proof trees (above) in Gontier et al. (2020). BOTTOM: An example ground constraint, which we use for analyzing consistency.

4.1 Relational Reasoning

CLUTRR Sinha et al. (2019) focuses on QA over synthetic stories about family relations as shown in Figure 3, and has more recently been extended to focus on proof generation (Gontier et al., 2020). As illustrated in Figure 5, we use the proof annotations in the latter work to generate intermediate propositions that track the time-course of family relations as they emerge at each new sentence.

Relying on the *clean* subset of CLUTRR stories Sinha et al. (2019) and proof annotations, breakpoints are added after each sentence. Propositional renderings of the explicit story facts, as well as intermediate propositions revealed in the proof annotations, were then added to each corresponding breakpoint in the story and serve as the *base* proposition set. From these base propositions, additional propositions, including negative and unknown propositions, were added using the following general constraints: **monotonicity**, that beliefs, once established to be true /false, cannot change; the **mutually exclusivity** of certain relations (e.g., *X is the grandfather of Y* is mutually exclusive with *X is the grandmother of Y*); **inverse relations** between certain relations (*e.g.,* that X_{fem} *is a sister of* Y_{fem} means that Y is a sister of X), and that all **non-deductively valid** propositions are unknown (i.e., with label **U**).[5] Such *ground* propositions con-

[5]We note that all such constraints remain faithful to the semantics of the original tasks, such as CLUTRR.

straints are included in the breakpoint annotations as symbolic expressions (see again Figure 5) to allow for measuring model consistency at inference time (later in Figure 6. See details in § A.3.

o.o.d evaluation. Stories in CLUTRR are characterized by their length k (number of events) and **generalization** testing is usually performed to measure generalization. Our main datasets (later seen in Table 1) consists of 13k training stories drawn from stories from k =2,..,5. We tune our models and evaluate on a mixture of in-domain and generalization stories of lengths k=2,..8 each containing around 1.5k stories (containing (avg) 10 propositions per breakpoint and 15 constraints per story). While these splits deviate from standard uses of CLUTRR, we also compare against standard splits (i.e., training on $k = 2, 3$ and testing on $k' = 2, ..10$) to look at the ability of training joint belief prediction and QA models on the original *QA task*.

4.2 Story Understanding

We experiment with the bAbI QA benchmark (Weston et al., 2016), which contains questions over stories about agents in controlled micro-worlds (see Figure 3). As with CLUTRR, the synthetic nature of domain makes it possible to automatically extract proposition annotations that express *object location* (e.g., *PersonX/ObjectX' is in Y*), *object possession* (*PersonX has ObjectY*), abstractions of *event post-conditions* (e.g., *PersonX took something* for the event *PersonX grabbed the ball*) and *pronoun* references (e.g., *He refers to John*). We use the Dyna-bAbI task generator (Tamari et al., 2022) to generate initial base propositions and, similar with CLUTRR, heuristically add more propositions using domain constraints (see § A.2 for more details).[6] We use propositional versions of the 7-task set introduced in Tamari et al. (2022). We specifically use the *long-form* version of this set, where stories all contain 20 events/breakpoints, and train on 500 examples per task (totaling 3.5k+1.4k training/evaluation stories, with an average of 10 propositions per breakpoint and 123 constraints per story).

o.o.d evaluation. In addition to training and testing on this set, we also look at joint training on proposition prediction and the original QA task. For evaluation we also consider a more challenging **hardQA** generalization task from Tamari et al. (2022), where the test set features compositions of concepts seen at training time. Appendix A.2 contains example inputs and further task details.

[6] In contrast to CLUTRR and TRIP, bAbI does not have explicit *unknown* proposition annotations, hence propositions either have label **E** or **C**.

4.3 Physical Commonsense Reasoning

We apply our approach to the recently introduced Tiered Reasoning for Intuitive Physics (TRIP) dataset (Storks et al., 2021). TRIP features a story plausibility end task, similar in scope to our proposition task, as well as a multi-tiered evaluation of models' reasoning process. Given a pair of highly similar human-authored short stories about everyday activities, models must jointly identify (1) the implausible story (**task1**) (2) a pair of conflicting sentences in the implausible story (**task2**) (3) the underlying physical states in those sentences causing the conflict (**task 3**). While **task3** takes the form of a breakpoint modeling task, where physical states are rendered as textual propositions, we model the first two tasks as text2text tasks using multi-task breakpoint models (details in the appendix and in § 5.1). We use the original splits, consisting of 675 plausible stories and 1472 implausible stories. While we focus on the multi-tiered evaluation, we devised a small *filtered* dev set (644 stories) for later model analysis (Table 5).

5 Modeling Details and Metrics

Here we detail our main breakpoint transformer (§5.1) following the framework in § 3 and all metrics used in our experiments (§ 5.2).

5.1 Modeling

Encoder We experimented with the T5 model (Raffel et al., 2020) using the implementation from Wolf et al. (2020). T5's bi-directional encoder was used for both our story encoder $\mathbf{enc}_{\text{story}}$ and proposition encoder $\mathbf{enc}_{\text{prop}}$. While any comparable encoder would suffice, we chose T5 due its common use in NLU and ability to perform generation, which we used to implement other components in the multi-task models discussed below. For efficiency reasons, we experimented with a combination of the smaller **T5-base** model (with 220M parameters) for datasets with long stories and many propositions (TRIP, bAbI) and **T5-large** (with 770M parameters) for CLUTRR.

Breakpoint and Proposition Embeddings For each story, individual breakpoint representations are first pooled from the **[B]** token hidden states in

the story encodings $\mathbf{c}_s$ (see again Figure 4). Following Ni et al. (2022), a linear projection and L2 normalization is applied to each representation to construct initial breakpoint embeddings. To allow for information transfer between different breakpoints, we then apply an additional self-attention layer (**sit-self**) over these resulting representations to obtain a *self-attention* breakpoint representation (see Fan et al. (2020) for a similar idea), which gets concatenated with the initial representation to create the final breakpoint embedding. Operationally, the self-attention layer takes the form of a standard transformer block (Vaswani et al., 2017) with a single attention head.

One subtlety in using a standard bi-directional encoder such as T5 is that each breakpoint token can look at future parts of the story. While the content of a breakpoint is often determined by the preceding sentence, in some cases it is important to have information about the future to obtain an accurate representation. For example, for the story *John has the apple.**[B]*$_1$ *He then moved to the kitchen **[B]***$_2$, knowing that *John* can't be *in the kitchen* at **[B]**$_1$ (a *pre-condition* of *move* events) requires looking into the future. To limit the amount of future information in part of our breakpoint representations, however, future masking is applied in the breakpoint self-attention layer described above.

To obtain a proposition embedding, we use the same T5 encoder over each text proposition prefixed with a special token, then take the hidden state of the target proposition. A final proposition representation is then similarly obtained using the same linear projection and normalization layers.

Proposition Classifier As in Li et al. (2021), we use a bilinear layer for proposition classification (score$(\cdot)$). Using the notation from § 3.3, probabilities $\hat{\mathbf{y}}(\mathbf{b}_j, \mathbf{p}) = \langle \Pr[\mathbf{E}(b_j, p)], \Pr[\mathbf{C}(b_j, p)], \Pr[\mathbf{U}(b_j, p)] \rangle$ for the 3 truth values of a proposition p are computed in the following way using the final breakpoint representation $\mathbf{b}_j$ and proposition encoding $\mathbf{c}_p$:

$$\text{score}(b_j, p) = \mathbf{b}_j^T \cdot \mathbf{M} \cdot \mathbf{c}_p + \mathbf{a}$$
$$\hat{\mathbf{y}}(b_j, p) = \mathbf{softmax}(\text{score}(b_j, p)).$$

Learning In addition to optimizing for the objective described in § 3.3 ($\mathcal{L}_{\text{prop}}$), we also experiment with multi-task models trained to do generation ($\mathcal{L}_{gen}$) and QA ($\mathcal{L}_{qa}$), both of which are formulated as text2text tasks and optimized using standard cross-entropy-based training. In the former case, we investigate two analogues to the unsupervised *denoising* objectives from (Raffel et al., 2020), which aim to increase the amount of local information contained in breakpoint representations.

The first is an **event generation** task that involves generating randomly chosen events from their right-most breakpoint encodings (e.g., generating the text *Susan's mother is Janice* from the encoding of **[B]**$_2$ in Figure 3). The second, which is inspired by Gontier et al. (2022), generates textual **abstractions** either of random events from breakpoints (in the case of TRIP, e.g., generating the abstracted text *PERSON dropped his OBJ...* from **[B]**$_1$ in Figure 3) or random pairs of events in a story (e.g., generating the text *A person received an apple* from the an encoding averaged from the two breakpoints **[B]**$_2$ and **[B]**$_3$ in Figure 3) (see additional details in § B.2).

Taken together, our full multi-task model's loss is: $\mathcal{L} = \lambda_1 \mathcal{L}_{\text{prop}} + \lambda_2 \mathcal{L}_{\text{qa}} + \lambda_3 \mathcal{L}_{\text{gen}}$ where $\lambda_{\{1,2,3\}}$ are task weights manually tuned during training. We used ADAM as our optimizer (Kingma and Ba, 2014). Standardly, hyper-parameter tuning and model selection was performed via a random search search in the style of Devlin et al. (2019) on held-out dev sets (see details in § B.1). Unless stated otherwise, we report the average of three random restarts for all models and their standard deviations.

Baselines We compare against two standard sentence representation learning approaches based on transformers and LSTMs. For the former we use the sentence transformer approach (Reimers and Gurevych, 2019) applied to our task, and for the latter we use a model close to Conneau et al. (2017). The set up is standard: stories and propositions are encoded separately using a single encoder and collected via mean (transformer) and max (BILSTM) pooling then aggregated via concatenation (in the style of InferSent Conneau et al. (2017)) and fed into a softmax classifier to make a belief prediction. Importantly, these baselines models are much less efficient compared with our *single read* breakpoint model, in that they require making multiple (**multi-pass late interaction**) forward passes through stories to create intermediate representations as illustrated in Figure 4. For the transformer models, with use the same T5 encoder as in the breakpoint models throughout all experiments.[7]

[7] As an additional check, we trained T5-based *proposition-only* baseline, similar to the *partial-input* baselines in NLI (Poliak et al., 2018), that make truth predictions from propositions *alone* to check for spurious patterns. These always

Given that our breakpoint models take full story texts as input, to make the baselines fully comparable, we similarly feed in the full story on each read with a similar special token (#) to mark the target intermediate point (e.g., In the story *John went to the store. He bought an apple* we feed the text *John went to the store. # He bought an apple* when modeling the first breakpoint).

Joint Modeling For CLUTRR and bAbI, we also compare our multi-task breakpoint model trained for QA against T5 and Bart (Lewis et al., 2020), both fine-tuned solely for QA.

5.2 Metrics

For proposition prediction tasks we measure overall **proposition accuracy** (%). Similarly for QA experiments, we follow other work in measuring exact match **EM accuracy** (%) against a model's generated output. For some of our analysis on CLUTRR (Figure 5), we measure the consistency of belief prediction using the **global consistency** metric ρ from Li et al. (2019), which measures the fraction of stories containing one or more constraint violation using the constraint annotations described in § 4. For example, using the constraint on the bottom Figure 5, we first have the model make predictions about the constituent propositions (1. *Derick is the father of Lisa*, 2. *Qiana is the wife of Derick*. 3. etc..) and see if those predictions symbolically satisfy the constraint.

For TRIP, we follow exactly the 3-tiered evaluation of Storks et al. (2021). We calculate: **Plausibility (task 1):** % of instances where the implausible story was correctly identified. **Consistency (task 2):** % of correctly identified implausible stories where the conflicting sentences were correctly identified. **Verifiability (task 3)**: % of instances with correct plausibility/consistency predictions, where all relevant physical states are also identified.

6 Results and Discussion

We focus on the following questions: 1. *Can our main model effectively and efficiently solve our new belief* ***proposition prediction*** *tasks (introduced in § 4) and model intermediate state?* 2. *Can we effectively integrate our breakpoint model into* ***joint models*** *for solving more complex tasks?*

Proposition Prediction We found breakpoint models to be effective at our proposition prediction tasks, most notably improving on the transformer

perform worse than our BILSTM baselines.

Proposition Prediction	
Model	Dev / Test Set + (std) (Acc %)
Majority Baseline	44.60 / 41.60
BILSTM (Multi-pass)	60.36 / 58.59 (±0.24)
T5-large (Multi-pass)	81.41 / 81.94 (±0.17)
BPT-large	**85.16 / 85.24** (±0.34)

Question-answering, **dev / test** + (std), (**EM** Acc %)		
Model	i.i.d	**generalization**
FT-T5-base	99.00 / 99.78 (±0.19)	84.19 / **75.13** (±0.94)
FT-Bart-base	98.65 / 98.94 (±0.78)	83.21 / 70.42 (±1.23)
BPT-base	**99.24** / 99.75 (±0.19)	83.61 / 74.84 (±0.89)

Table 1: TOP: Proposition prediction results on CLUTRR on the main mix dev and test sets comparing our breakpoint model (**BPT**) with baselines. BOTTOM: Evaluation on standard CLUTRR QA ($k = 2, 3$) comparing our breakpoint model trained joint with QA to fined-tuned (**FT**) T5 and Bart models.

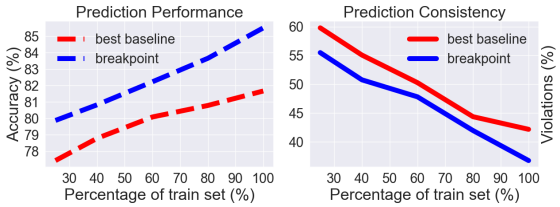


Figure 6: Effect of training data size on proposition prediction (left) and global consistency ρ (right, *lower is better*), on CLUTRR dev (best of 3 random runs).

Multi-pass baselines for CLUTRR prediction from 81.9 to 85.2 (top of Table 1, both an over 23% improvement over our BILSTM baseline, suggesting task difficulty). Based on the plots in Figure 6, we also found our models to be more efficient learners (e.g., achieving comparable performance to baselines using only 60% training data) and to exhibit less global constraint violations in the i.i.d setting (with a 6% reduction in constraint violations ρ), thus leading to more consistent belief states.

Model	**i.i.d**		**hard QA**
	Prop.%	**QA%**	**QA%**
Majority	65.87	–	–
FT-T5-base (QA)	–	97.29 (±0.14)	69.09 (±0.79)
FT-Bart-base	–	**97.57** (±0.31)	67.21 (±0.80)
BILSTM (Multi)	80.2 (±0.16)	–	–
T5-base (Multi)	**99.1** (±0.21)	–	–
BPT-base	98.5 (±0.10)	–	–
BPT-base + QA	98.5 (±0.10)	94.9 (±0.60)	**70.51** (±0.29)

Table 2: bAbI proposition prediction (**Prop. %**) and **QA** performance on the main i.i.d and **hardQA** test sets.

For bAbI (Table 2) all transformer-based models achieve near perfect accuracy (and significantly outperform our BILSTM model); as such, models have near perfect consistency on the underlying constraints (not shown). Given that bAbI stories are considerably longer than CLUTRR stories

(each containing 20 events/breakpoints), these results show the feasibility of modeling long contexts with our model and representing complex state information with individual breakpoints. In contrast to the baseline transformers, here we also see considerable practical improvements in **training time efficiency** due to our *single read* architecture, resulting in a 54% reduction in training time (from around 63 hours for multi-pass models to around 34 for ours on a single RTX A6000 GPU).

CLUTRR i.i.d vs. generalization (**gener.**) splits		
	i.i.d ($k = 2, ..5$)	**gener.**
Baseline (best run)	94.54 ($\rho = 32.1$)	61.7 ($\rho = 96.2$)
BPT (best run)	95.69 ($\rho = 25.9$)	**69.2** ($\rho = 97.6$)

Table 3: Comparison between i.i.d and compositional settings for CLUTRR.

Our model's proposition prediction consistency is 7.5% higher than that of the baseline, in terms of the ρ metric reported in Table 3. As an important caveat, however, in absolute terms, even our breakpoint model has much lower consistency on generalizations tasks (69.2%) than in the i.i.d. setting (95.7%). We discuss this further in § 8.

Joint Training When trained jointly for both proposition prediction and QA, we found minimal to no impact on end-task performance, as shown on the bottom of Table 1 for CLUTRR and in Table 2 for bAbI (with a small improvement on the generalization QA task at the cost of a mere 2% degradation in i.i.d. QA performance). This shows the viability of integrating our belief tracking mechanism into existing transformer pipelines without significant performance drops. As first motivated in Figure 1, it also permits the development of more debuggable systems where the results of QA can be checked against the model's beliefs.

Split	**Model**	**Task 1** *(Plaus.)*	**Task 2** *(Consist.)*	**Task 3** *(Verif.)*
Dev	RoB	73.6	22.4	10.6
	BPT-base	**81.99**(±0.91)	**58.07**(±0.76)	**36.44**(±0.53)
Test	RoB	72.9	19.1	9.1
	BPT-base	**80.55**(±1.20)	**53.83**(±1.65)	**32.37**(±0.27)

Table 4: Results on the TRIP 3-tiered physical commonsense reasoning benchmark, our main **breakpoint** model (**BPT**) compared against the RoBERTa-based approach (RoB) of Storks et al. (2021).

Through our results on TRIP (Table 4), we also see the viability of adding our belief tracking mechanism into more complex modeling pipelines. We were specifically able to obtain SOTA performance on this task and outperform the larger and highly tailored task-specific model architecture based on RoBERTa-large used by Storks et al. (2021).

CLUTRR (mix dev)		
	Prop. Acc%	Global Violations ρ
BPT-large (best run)	85.5 (Δ)	36.7 (Δ)
- brk self-attn	77.3 (-8.12)	54.3 (-17.61)
- event generation	82.1 (-3.36)	41.8 (-5.07)
- abstraction	82.1 (-3.35)	42.8 (-6.06)
BPT-base	81.8 (-3.62)	44.3 (-7.61)
TRIP (fitered dev)		
BPT-base (best run)	92.8 (Δ)	–
- brk self-attn	89.43 (- 3.36)	–
- event generation	89.43 (- 3.36)	–
- abstraction	92.9 (+0.10)	–

Table 5: Breakpoint model feature ablations.

Additional Analysis We see in Table 5 for CLUTRR that having an additional self-attention aggregation layer when constructing breakpoint representations (**-brk self-attn**, § 5.1) is very important for accuracy and consistency (we find similar results for TRIP, bottom). This suggests that further improvements might be achieved through improved pooling and masking strategies for constructing breakpoint representations. We also see the advantages of having auxiliary generation losses (**event generation, abstraction**) for improving accuracy and performance.

7 Conclusion

Being able to track the beliefs of models remains a formidable challenge at the forefront of model interpretability. In this paper, we presented a new representation learning framework, *breakpoint modeling*, that facilitates end-to-end learning and tracking of beliefs at intermediate states in narrative text. On a diverse set of NLU tasks, we show the benefit of our approach (based on T5) over conventional learning approaches in terms of improved belief prediction performance on new belief tracking tasks and processing efficiency. We also show the feasibility of recasting existing tasks into our framework and integrating our approach into existing transformer-based NLU pipelines, which we believe can help to improve the interpretability of these models as part of this larger challenge.

Acknowledgements

The authors thank the Aristo team for valuable feedback. This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 852686, SIAM, Shahaf). Part of this research was also supported by the European Research Council, ERC-StG grant no. 677352 (Tsarfaty), which is gratefully acknowledged.

8 Limitations

Below we summarize the main limitations of our current breakpoint models and the techniques pursued in this study.

Compositional Generalization Despite richer supervision over intermediate states, compositional generalization performance remains a significant challenge (on bAbI and CLUTRR generalization splits, see §6) for future work, which shows that our approach inherits many of the limitations in the generalization ability of large-scale LMs more broadly. Following Kim et al. (2021) and others, we hypothesize that the all-to-all attention employed by Transformers in creating token encodings (including the breakpoint tokens) is a factor in non-compositional behavior; such attention is more vulnerable to overfitting spurious patterns. Accordingly, more advanced attention masking (Kim et al., 2021) and supervision (Yin et al., 2021) approaches are promising directions to explore.

Our notion of "belief" While breakpoints provide an indication of intermediate model "beliefs", they are also different from beliefs in important ways. In particular, the causal relation between information represented in breakpoints and generated model outputs is unclear (see also Li et al. (2021) for similar caveats in standard NLMs). For example, models may generate outputs that are inconsistent with their own breakpoint belief states. Interestingly, breakpoint models may offer new ways to address these limitations by more explicitly representing intermediate reasoning steps; neural logic losses (Li et al., 2019) can help enforce belief consistency between sets of propositions (§3.3).

Task and domain limitations Finally, our experiments are still limited to datasets involving relatively short (TRIP) and synthetic (bAbI, CLUTRR) inputs with limited semantics. Further work is needed to address more natural and complex language to ultimately develop more robust breakpoint models. In contrast to standard end-to-end QA methods, breakpoint modeling requires more costly annotation, as training currently requires some form of supervision on intermediate states, beyond the final target output. Thus, developing new methods for collecting such annotations with minimal engineering effort remains a challenge.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554, San Diego, California. Association for Computational Linguistics.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction through search over statement compositions. *arXiv preprint arXiv:2201.06028*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. 2020. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*.

Stefan L. Frank, Mathieu Koppen, Leo G.M. Noordman, and Wietske Vonk. 2003. Modeling knowledge-based inferences in story comprehension. *Cognitive Science*, 27(6):875–910.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah D Goodman, and Christopher Potts. 2021. Inducing causal structure for interpretable neural networks. *arXiv preprint arXiv:2112.00826*.

Richard M. Golden and David E. Rumelhart. 1993. A parallel distributed processing model of story comprehension and recall. *Discourse Processes*, 16(3):203–237.

Nicolas Gontier, Siva Reddy, and Christopher Pal. 2022. Does entity abstraction help generative transformers reason? *arXiv preprint arXiv:2201.01787*.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Chris Pal. 2020. Measuring systematic generalization in neural proof generation with transformers. In *Advances in Neural Information Processing Systems*, volume 33, page 22231–22242. Curran Associates, Inc.

Andrew R Haas. 1987. The case for domain-specific frame axioms. In *The Frame Problem in Artificial Intelligence*, pages 343–348. Elsevier.

Austin W Hanjie, Ameet Deshpande, and Karthik Narasimhan. 2022. Semantic supervision: Enabling generalization over output spaces. *arXiv preprint arXiv:2202.13100*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. 2021. BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8849–8861, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1264–1279, Online. Association for Computational Linguistics.

Juyong Kim, Pradeep Ravikumar, Joshua Ainslie, and Santiago Ontanon. 2021. Improving compositional generalization in classification tasks via structure annotations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–645, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Fred Landman. 2012. *Structures for semantics*, volume 45. Springer Science & Business Media.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. 2019. A logic-driven framework for consistency of neural models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3924–3935, Hong Kong, China. Association for Computational Linguistics.

Shih-Ting Lin, Ashish Sabharwal, and Tushar Khot. 2021. ReadOnce transformers: Reusable representations of text for transformers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7129–7141, Online. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8713–8721.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. 2021. Tiered reasoning for intuitive physics: Toward verifiable commonsense language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4902–4918, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Ronen Tamari, Kyle Richardson, Noam Kahlon, Aviad Sar-shalom, Nelson F. Liu, Reut Tsarfaty, and Dafna Shahaf. 2022. Dyna-bAbI: unlocking bAbI's potential with dynamic synthetic benchmarking. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 101–122, Seattle, Washington. Association for Computational Linguistics.

Ronen Tamari, Chen Shani, Tom Hope, Miriam R L Petruck, Omri Abend, and Dafna Shahaf. 2020. Language (re)modelling: Towards embodied language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6268–6281, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Noortje J. Venhuizen, Matthew W. Crocker, and Harm Brouwer. 2019. Expectation-based comprehension: Modeling the interaction of world knowledge and linguistic experience. *Discourse Processes*, 56(3):229–255.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable natural language processing. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

A Dataset Details

In this section, we provide additional details about all datasets.

A.1 TRIP

As described in §4, the TRIP benchmark consists of 3 tiered tasks: (1) plausibility (2) consistency (3) verifiability. To apply our model to TRIP, we convert the first two tasks to a text2text format: the first task involves taking two stories `(A) storyA (B) storyB $plaus` as input text and producing a text label {A,B} to identify the implausible story; task 2 involves taking a labeled story `sentence1 1 sentence2 2.. $conflict` and generating the labels identifying the problematic sentences.[8] We convert the third task to breakpoint format by converting state change labels to textual propositions associated with the corresponding timesteps. Figure 8 shows an example instance from the TRIP development set. Note that each task is effectively rendered as two instances: the first instance addresses task 1 (as QA), and the second jointly addresses tasks 2 (QA) and 3 (proposition prediction).

State changes in TRIP are defined either as effects or preconditions (Storks et al., 2021) and this information must be preserved in the conversion to breakpoint format. Preconditions are propositions that hold *before* a described event; for example, the proposition "oven was open" should be **true** before the sentence "John closed the oven." Effect propositions are propositions that hold after a described event; the proposition "oven is open" should be **false** after "John closed the oven." We represent precondition and effect propositions simply by modifying the proposition tense. Given breakpoint b_t, for associated precondition propositions at time t, we use past tense ("oven was open"). For effect propositions at time t, we use present tense ("oven is open").

While the TRIP data includes state information for all time steps and entities, we follow the official evaluation procedure[9] and only score the subset of state changes defined to be relevant in the pair of conflict sentences. At training time, we use all available state change information for training.

Finally, while most state changes in TRIP are attributes that can be **true**, **false** or **unknown** (and thus can be directly converted to proposition form), location attributes are formulated as k-way classification problems. For example, an object location attribute change is represented by 1 of 9 possible classes (see Table 5 in Storks et al. (2021) and blue propositions in Fig. 8). To facilitate equivalent evaluation of k-class predictions with our breakpoint model, we consider the predicted **true** score for each of the possible k propositions and take the maximum scoring proposition to be the predicted value.[10]

A.2 bAbI

A.2.1 Proposition generation

As detailed in § 4, base propositions for bAbI are generated using the Dyna-bAbI tool (Tamari et al., 2022). From this, new propositions are derived from the following general constraints: **location/-possession uniqueness** that dictate that objects can only be in one location at a time and possessed by a single agent (e.g., *John* cannot simultaneously be *in the kitchen and living room*), **mutually exclusivity** between event types (e.g., that *dropping a ball* is the opposite of *picking up a ball*); **explanation frame rules** (Haas, 1987) that dictate that objects, when left unchanged, maintain their location and their possession through time (e.g., *John is in the kitchen* or *John has the apple* stays true until there is an explicit event that changes this).

A.2.2 Task details

The training data includes 500 samples per task type, where the tasks follow the same structure as the *concat(T7)* dataset described in (Tamari et al., 2022) (Table 6 in that work), with the only difference being the story length which was fixed to 20 sentences to match the test data. The **hardQA** generalization task was generated using the same settings as the *mix(T7)* evaluation set from (Tamari et al., 2022), including the same 3 question types with 1,000 samples for each type (also Table 6 in Tamari et al. (2022)). Figure 7 shows example stories from the training and hardQA test splits.

A.3 CLUTRR

We note that all of the underlying story data was generated from scratch and relies on the publicly available task generators from Sinha et al. (2019)

[8] `$plaus` and `$conflict` are special tokens that prompt the model to output an answer for tasks 1 and 2, respectively.

[9] https://github.com/sled-group/Verifiable-Coherent-NLU/blob/main/Verifiable-Coherent-NLU.ipynb

[10] Inspired by a similar method in Li et al. (2021).

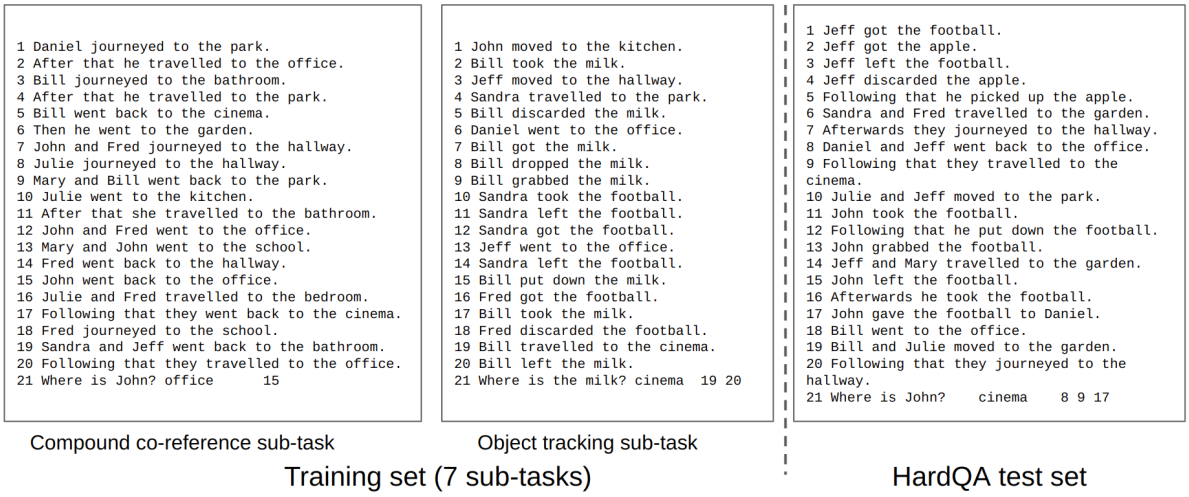


Figure 7: Examples from the bAbI story understanding task. The train set includes 7 sub-tasks, such as co-reference and object tracking (left). The **hardQA** sample (right) incorporates novel compositions of concepts seen separately at training time. Beyond the question answering task, each example also includes proposition prediction at each time step (not shown here, see Figure 1) for example.

and Gontier et al. (2020)[11]. As detailed in Gontier et al. (2020), leakage among the proofs and propositions in stories of the same k can be a problem. Using some of their ideas, we avoided this by expanding the inventory of names used in training and abstracted names for parts of the training. We verified the hardness of our data by training a no-story proposition-only baseline an found it to have low performance, and also manually verified all inference rules used for generating propositions.

B Training details

B.1 Hyper-parameters

All hyper-parameter tuning for our main models was performed via a random search in the style of Devlin et al. (2019). Model selection was performed by selecting models with the highest validation accuracy for each task (e.g., proposition accuracy for our proposition tasks, exact match for the QA experiments). Unless noted otherwise, we report the average of models with the optimal hyper-parameters based on 3 random re-starts; early stopping was applied throughout. All experiments were performed on NVIDIA AX6000 GPU hardware on a single GPU.

breakpoint models: *learning rate* (we experimented in the range of `1e-3` to `5e-6`, we generally found `5e-5` to be optimal for most experiments), *number of epoch* (up to 35 for CLUTRR, TRIP and 150 for bAbI), *batch size* (in the range of *2* to *16*, memory permitting, we found 2 to be optimal for bAbI and TRIP experiments, and 4 for CLUTRR) and *weight decay* (set to `0.001`) and *warmup steps* (from `500` to `1k` steps). See the project repository for further details

joint models For multi-task training, parameters $\lambda_{\{1,2,3\}}$ were hand tuned, with λ_1 set to `1.0` for all proposition prediction tasks (with λ_2=`0.1` for most tasks). For joint QA tasks, we found setting $\lambda_1 = 1.0$ and $\lambda_1 = 1.0$ to be optimal, with an initial warmup before turning on the proposition prediction loss (usually between 5-10 epochs). Given the high cost of training the bAbI breakpoint QA model in Table 2, the joint QA + prop models described on the last row start training from the **BPT-base** checkpoints described in the row above.

B.2 Auxiliary Generation Losses

As detailed in § 5.1, we jointly trained our breakpoint models with additional generation losses that aim to mimic some of the unsupervised *denoising objectives* used in Raffel et al. (2020). Whereas in standard denoising you might try to generate from a text input **A dog <mask> while running** the output text <*mask*> *barked loudly* <*mask*>, from an original text *A dog barked loudly while running* (with full attention over the input text), in our case we try to generate from a story **John went to the store [B]$_1$ He then picked up the**

[11] See full details at: https://github.com/facebookresearch/clutrr and https://github.com/NicolasAG/SGinPG

```
# Task 1 (plausibility)
{
        "example id": "414-C0-a",
        "question": "(A) John turned on the oven [B] John put the cake in the oven [
           B] John got the ice cream out [B] John put some ice cream in a red bowl
           [B] John put the red bowl in the oven [B] (B) John turned on the oven [B
           ] John put the cake in the oven [B] John got the ice cream out [B] John
           put some ice cream in a red bowl [B] John put the rest of the ice cream
           in the fridge [B] $plaus",
        "answer": "B"
}

# Tasks 2 + 3 (consistency + verifiability)
{
        "example id": "414-C0-b",
        "question": "John turned on the oven 0 [B] John put the cake in the oven 1 [
           B] John got the ice cream out 2 [B] John put some ice cream in a red
           bowl 3 [B] John put the red bowl in the oven 4 [B] $conflict",
        "answer": "3,4"
        "proposition_lists": [
                [...], # sent. idx 0
                [...], # sent. idx 1
                [...], # sent. idx 2
                [
          "red bowl is occupied",
          "ice cream is put into a container",
          "ice cream does not move to a new location",
          "ice cream disappears",
          "ice cream is picked up",
          "ice cream is put down",
          "ice cream is put on", "ice cream is removed",
          "ice cream is taken out of a container",
          "ice cream moved somewhere new",...
                ], # sent. idx 3
                [
        "red bowl is put into a container", "oven was powered",
        "oven was open", "oven was turned on",...
            ], # sent. idx 4
                ],
        "labels": [
            [...], # sent. idx 0
            [...], # sent. idx 1
            [...], # sent. idx 2
            ["true", "true", "false", "false", "false", "false", "false", "false","
               false", "false",...], # sent. idx 3
            ["true", "true","true", "true",...], # sent. idx 4
        ]
}
```

Figure 8: Rendering of TRIP instance in breakpoint format. Breakpoint models can operate in standard text-to-text mode, generating output answers in response to questions, and additionally they can provide joint predictions over propositions associated with each sentence. Propositions in blue indicate *location* attributes which are evaluated as k-class predictions. See Appendix A.1 for further details on instance construction.

apple [B]$_2$ the raw event text *John went to the store* from the corresponding raw breakpoint hidden state for the special token **[B]**$_1$ alone. In addition to this *event generation* task, we also experimented with a *abstraction* generation task: given two stories in a batch and two random breakpoints within those stories, e.g., **John went to the kitchen [B]**$_{1,1}$**...** and **Mary went to the kitchen [B]**$_{2,1}$**..**, we ask the model to generate an abstract textual description of the two events only from the mean of the two breakpoint hidden states, i.e., $\texttt{abstraction}(\textbf{[B]}_{1,1}, \textbf{[B]}_{2,1}) =$ *A person went to the kitchen*. (This was inspired by the abstraction generation ideas from Gontier et al. (2022)).

During training, both forms of generation were done by randomly selecting a single breakpoint example and abstraction pair for each story in the batch and computing a standard loss over the generated texts and abstractions. Using symbolic annotations of both the CLUTRR and bAbI training events, a deterministic algorithm was implemented for creating abstracted texts on the fly for training. For TRIP, where logical annotations are not available, the abstraction task was replaced by the task of generating versions of text replaced with POS tags (e.g., *John turned off the stove* would be turned into *PER turned off the NOUN*).