Breakpoint Transformer Model For TRIP (Tiered Reasoning for Intuitive Physics)

Kyle Richardson, Ronen Tamari, Oren Sultan

Allen Institute for AI(AI2),
The Hebrew University of Jerusalem(HUJI)

Agenda

slides

- Overview for the Breakpoint Modeling framework & Breakpoint Transformer Model (3-10)

- Overview for this work (11)
- Overview for the TRIP dataset and the proposed tasks (12-18)
- TRIP data conversion for the Breakpoint Model (19-27)
- Experiment: applying Breakpoint Model to TRIP (28-30)
- Breakpoint Model evaluation results (31-39)

Overview for The Breakpoint Modeling framework

Overview for The Breakpoint Modeling framework

- **The goal:** teach models to track their beliefs through intermediate points in text
  - Hopefully to achieve **better performance**, particularly for **narrative understanding**.
  - **Model probing - reason transparency**, which contributes to model reliability.
  - **Complex declarative constraints**.

- **The solution:** given any encoder and data marked with intermediate states(**breakpoints**) along with textual **propositions**, and symbolic constraints that hold between propositions, models are trained to build intermediate representations that facilitate correct and consistent predictions, along with solving their ordinary set of tasks.

- **Implementation**: Current implementation for this framework is the Breakpoint Transformer that builds on the pretrained **T5 Transformer Model**.

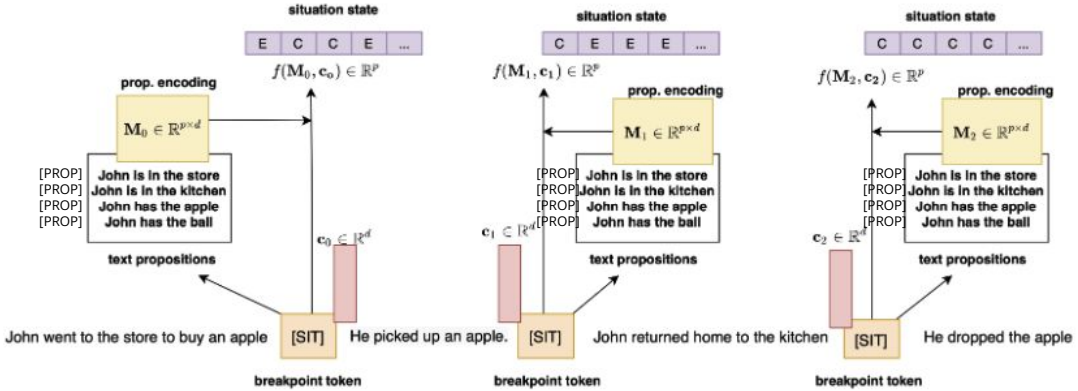
- **Results**: achieves better performance on benchmark tasks for **story understanding** and **relation reasoning**.

Examples for stories and breakpoint propositions

Task	**Example Stories**	**Breakpoint Propositions**
Relational Reasoning (CLUTRR)	John is the brother of Susan [SIT]1 Susan's mother is Janice [SIT]2...	P1: { 'Susan is the sister of John' true, 'Susan is the sister-in-law of Janice' false,...} P2: { 'Janice is the mother of John' true, 'John is the father of Janice' false, …}
Story Understanding (bAbi)	John moved to the kitchen [SIT]1 He picked up an apple [SIT]2 John then gave the apple to Mary [SIT]3	P1: { 'John has the apple' false, 'John is in the kitchen' true,...} P2: { 'John has the apple' true,…} P3: { 'John has the apple' false, 'Mary has the apple' true, … }

- The breakpoint propositions are used as supervision at training time or as queries to perform model probing.
- When coupled with predictions, they constitute the beliefs of the model.

Breakpoint Model applied to story understanding task



Breakpoint and proposition encoding

- **[SIT]** - special token which is used to delimit the boundary of each breakpoint. It represents all of the tokens that came before(all of the information in the story up to this point). Each [SIT] token is aligned with a set of text propositions.

- There are two encoders: **enc_story**, **enc_prop** that are used to generate representation **c** for each breakpoint in the story and each proposition, respectively.

- Representations of breakpoints $\mathbf{c}_{\mathbf{SIT}} \in \mathbb{R}^d$ are pooled from a single encoding of an input story: $\mathbf{c}_s \leftarrow \mathbf{enc}_{\text{story}}(s) \in \mathbb{R}^{|s| \times d}$

- Similarly for the representations of propositions $\mathbf{c}_{\text{prop}} \in \mathbb{R}^d$ (special token [PROP]), by using $\mathbf{enc}_{\text{prop}}$

- The **bi-directional encoder** of **T5-Large Model** has been used for both **enc_story**, **enc_prop**
  - One consequence of using bi-directional encoder is that each token as access to the full story (including the tokens that come later)

Proposition scorer classifier

- Given a breakpoint encoding and an aligned proposition encoding , it makes the prediction about a proposition at that breakpoint {false, unknown, true} -

  Bilinear scorer- computes score as bilinear transformation between situation vector and corresponding proposition vector.

$$score(s_j, p).shape \rightarrow 3 \qquad M.shape \rightarrow d, d, 3$$

$$\text{score}(s_j, p) = \mathbf{c}_{\mathbf{[SIT]^{self}_j}}{}^T \cdot \mathbf{M} \cdot \mathbf{c}_p + \mathbf{b}$$

$$\mathbf{y}^*(s_j, p) \sim \mathbf{X}_{s_j,p} = \mathbf{softmax}(\text{score}(s_j, p))$$

- **Notations:** E(s_j, p), C(s_j, p), U(s_j, p) - breakpoint s_j entails/ contradicts or has an unknown relation to proposition p.

```
torch.nn.functional.bilinear(input1, input2, weight, bias=None)
```

Applies a bilinear transformation to the incoming data: $y = x_1^T A x_2 + b$

Shape:

- input1: $(N, *, H_{in1})$ where $H_{in1} = \text{in1\_features}$ and $*$ means any number of additional dimensions. All but the last dimension of the inputs should be the same.
- input2: $(N, *, H_{in2})$ where $H_{in2} = \text{in2\_features}$
- weight: $(\text{out\_features}, \text{in1\_features}, \text{in2\_features})$
- bias: (out_features)
- output: $(N, *, H_{out})$ where $H_{out} = \text{out\_features}$ and all but the last dimension are the same shape as the input.

Learning - logical objective to different loss functions

Given a dataset D of n stories each containing m breakpoints paired with labeled propositions, gold label is $Y^* \in \{E, C, U\}$

The goal of the model: to make the correct truth assignments to propositions and hence have beliefs that align with D. Logically:

$$\forall s^{(i)} \in D : \wedge_{s_j \in s^{(i)}} \wedge_{p \in s_j} true \rightarrow Y^*(s_j, p)$$

By using $y^*(s_j, p) \sim X_{s_j,p}$ to denote the model's probability corresponding to proposition's target label $p(Y^*(s_j, p))$ we can translate this logical objective to the following loss:

$$\mathcal{L}_{prop} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{p \in s_j} -log y^*(s_j, p)$$

The advantage - express more complex forms of objectives and constraints

- Constraints that are based on symbolic knowledge involving propositions within and across breakpoints. For example, the following symbolic constraints:
  Logically: $Y^*(s_j, p) \rightarrow Y^{*'}(s_{j'}, p')$

$$\forall s^{(i)} \in D : \wedge_{s_j \in s^{(i)}} Y^*(s_j, p) \rightarrow Y^{*'}(s_{j'}, p')$$

Name	Boolean Logic	Product	Gödel	Łukasiewicz
Negation	$\neg A$	$1-a$	$1-a$	$1-a$
T-norm	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
T-conorm	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Residuum	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1, \text{ if b} \geq \text{a}, \\ b, \text{ else} \end{cases}$	$\min(1, 1-a+b)$

For example:
“John as the Apple”, true → “Mary as the Apple”, false
“Susan as the Apple”, false ,…. (For other entities)

Overview for this work

- **The goal:** Apply the Breakpoint Model to new kind of interesting datasets.
- **Duration**: September 2021 - December 2021(including).
- **Propara dataset** (procedural texts understanding) - September
  - Data conversion for the model
  - Experiments
  - Evaluation
- **TRIP dataset** (story understanding- common sense AI) - October - December
  - Data conversion for the model
  - Extension of the model for precondition and effect
  - Experiments
  - Evaluation

Overview for the TRIP dataset and the proposed tasks

The TRIP dataset

- **Motivation for creating the dataset:** to evaluate the reasoning and the deep understanding of the tasks by models, instead of only the end performance.

- TRIP is a novel physical common sense reasoning dataset, the end-task is story plausibility classification. Notably, however, it includes dense annotations for each story capturing **multiple tiers of reasoning**, beyond the end-task.

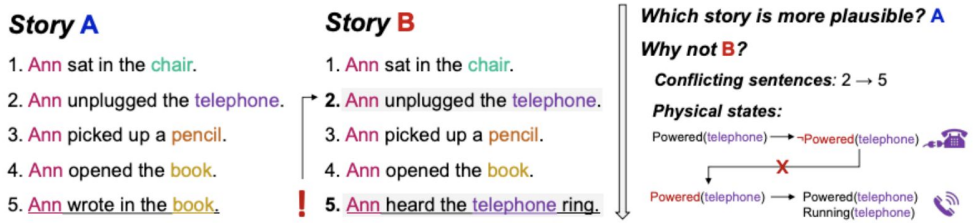
The three proposed tasks

1. **Story Classification(the end-task)-** determines which story is plausible

2. **Conflict Detection-** identifying conflict sentences in the form Si->Sj. While Sj is the **breakpoint** and Si is the **evidence**.

3. **Physical state classification-** two tasks: **precondition** and **effect** classification. **for example:** “John cut the cooked potato”

- Plausible stories were crowd-sourced from Amazon Mechanical Turk. To convert each story into several implausible stories, they hired separate workers, each wrote a new sentence to replace a sentence in the original story.



Statistics on the TRIP dataset

Measure	Train	Dev	Test	All
# plausible stories	370	152	153	675
# implausible stories	799	322	351*	1472
avg. # sentences	5.1	5.0	5.1	5.1
avg. sentence length	8.3	8.0	8.5	8.3
avg. # conflicting sentence pairs	1.2	1.2	1.2	1.2

Physical annotation label space

They defined a space of 20 physical attributes which capture most conflicts found in the stories.

- For **human** they track 5 physical attributes: location, hygiene and whether a human is conscious, dressed or wet.
- For **objects** they track 15 physical attributes: exists, is clean, connected to power, functional, in pieces, etc..

Label	Human Location	Object Location	Other Attributes
0	irrelevant	irrelevant	irrelevant
1	disappeared	disappeared	*false* → *false*
2	moved	picked up	*true* → *true*
3	–	put down	*true* → *false*
4	–	put on	*false* → *true*
5	–	removed	___ → *no*
6	–	put in container	___ → *true*
7	–	taken out of container	*false* → ___
8	–	moved	*true* → ___

The results of large LMs on the tasks

while large LMs can achieve high end-task performance(up to 78% accuracy), they struggle to jointly support their predictions with the proper evidence (only up to 11% of examples supported with correct physical states and conflicting sentences).

TRIP data conversion for the Breakpoint Model

Data conversion running arguments

- **Dataset_type** - train/dev/test

- **Both_implausible_plausible** - 0=conversion for implausible stories only, 1=also for plausible stories.
  We didn't find a significant difference in performance when we trained on plausible stories as well.

- **Unk_sampling** - for example: 0.2 means to sample 20% of unknown.
  Has an Impact on evaluation!

- **Unk_sampling_Seed** - we used 10 different seeds, because when we random sampling the unknown, we have variance for different runs with the same sampling ratio.
  - We can identify and restore the run by the seed.
  - We also show the mean of all 10 runs in development set evaluation

Data conversion running arguments

- **Pre_post_format-** 0=regular format and 1=pre post format(the extended model)
  We can achieve good performance also without the extension option.

- **Before_after** - for example, for the proposition “John is conscious”.
  - before_after=0: will create “John **was** conscious” and “John is conscious”
  - before_after=1: will create “**before**: John **was** conscious” and “John is conscious”
  - before_after=2: will create “**before**: John **was** conscious” and “**after**: John is conscious”

  There is no significant difference in performance between the options, so we stick with the first.

Considered but didn’t use

We considered to generate for every proposition, the opposite one.
For example: for the ‘conscious’ attribute [‘Tom’, 2] which means both precondition and effect are ‘true’. Our conversion will generate **“Tom was conscious”** and **“Tom is conscious” with the label ‘true’**. By using the negative word for ‘conscious’ we can double the amount of propositions and also create **“Tom was unconscious”**, **“Tom is unconscious”** both **with the label ‘false’**.
It does not improves the performance, we decided not to use it, because of the disadvantage of doubling the data.

```
att_adj = { 'conscious': ('unconscious', 'conscious'),
            'wearing': ('undressed', 'dressed'),
            'h_wet': ('dry', 'wet'),
            'hygiene': ('dirty', 'clean'),
            'exist': ('nonexistent', 'existent'),
            'clean': ('dirty', 'clean'),
            'power': ('unpowered', 'powered'),
            'functional': ('broken', 'functional'),
            'pieces': ('whole', 'in pieces'),
            'wet': ('dry', 'wet'),
            'open': ('closed', 'open'),
            'temperature': ('cold', 'hot'),
            'solid': ('fluid', 'solid'),
            'contain': ('empty', 'occupied'),
            'running': ('turned off', 'turned on'),
            'moveable': ('stuck', 'moveable'),
            'mixed': ('separated', 'mixed'),
            'edible': ('inedible', 'edible')}
```

Data conversion - important methods

- **Generate propositions for ‘location’ and ‘h_location’ attributes:** By reading the ‘state’ which gives an entity and the attribute label value, and by using ‘att_change_dir’ dictionary, we generate one proposition with true, and for other attributes values, with false.

  For example: [‘Tom’, 0] for h_location → proposition = (“Tom does not move to a new location”, true). In addition, we create two more propositions with the other ‘h_location’ attribute values, with false.

```
att_change_dir = {'h_location': {0: 'does not move to a new location', 1: 'disappears', 2: 'moves somewhere new'},
            'location': {0: 'does not move to a new location', 1: 'disappears', 2: 'is picked up', 3: 'is put down',
                         4: 'is put on', 5: 'is removed', 6: 'is put into a container',
                         7: 'is taken out of a container', 8: 'moved somewhere new'},
            'default': {0: (-1, -1), 1: (0, 0), 2: (1, 1), 3: (1, 0), 4: (0, 1), 5: (-1, 0),
                        6: (-1, 1), 7: (0, -1), 8: (1, -1)}}
```

-1=unknown, 0=false, 1=true

Data conversion - important methods

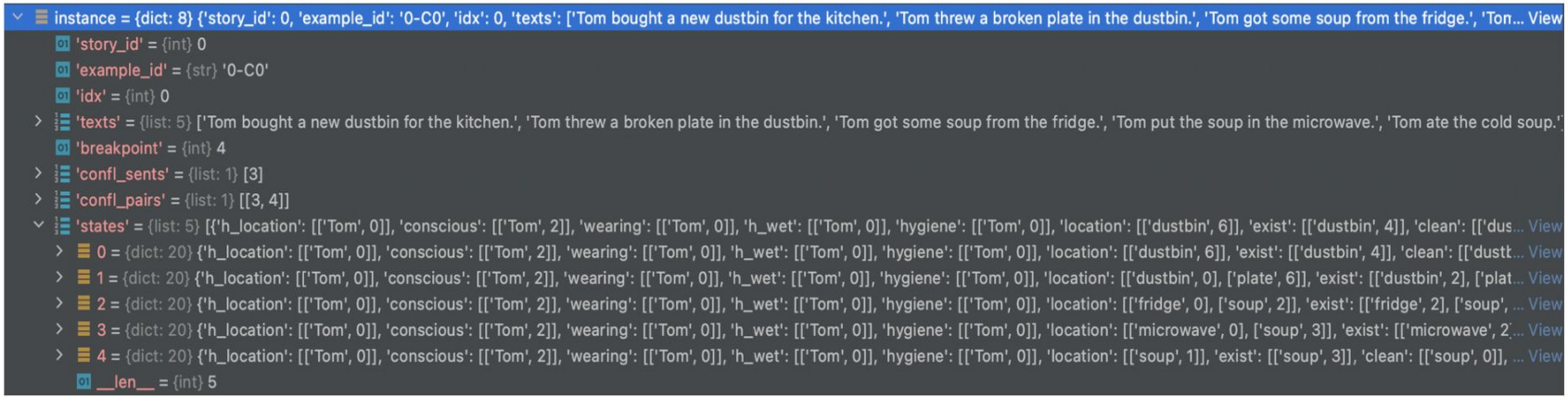
- **Generate precondition and effect propositions for other attributes:**
  - For the **precondition** propositions we will use ‘was’/’were’ instead of ‘is’/’are’.
  - If the value of the attribute is ‘unknown’, we will create the proposition w. Probability —unk_sampling.
  - If the attribute’s value is true or false, we will create the proposition by using the ‘att_change_dir’ dictionary.

  - **For example:** [‘Tom’, 2] for the attribute ‘conscious’ →
    (“Tom was conscious”, true) and (“Tom is conscious”, true)

```
att_change_dir = {'h_location': {0: 'does not move to a new location', 1: 'disappears', 2: 'moves somewhere new'},
            'location': {0: 'does not move to a new location', 1: 'disappears', 2: 'is picked up', 3: 'is put down',
                         4: 'is put on', 5: 'is removed', 6: 'is put into a container',
                         7: 'is taken out of a container', 8: 'moved somewhere new'},
            'default': {0: (-1, -1), 1: (0, 0), 2: (1, 1), 3: (1, 0), 4: (0, 1), 5: (-1, 0),
                        6: (-1, 1), 7: (0, -1), 8: (1, -1)}}
```

-1=unknown, 0=false, 1=true

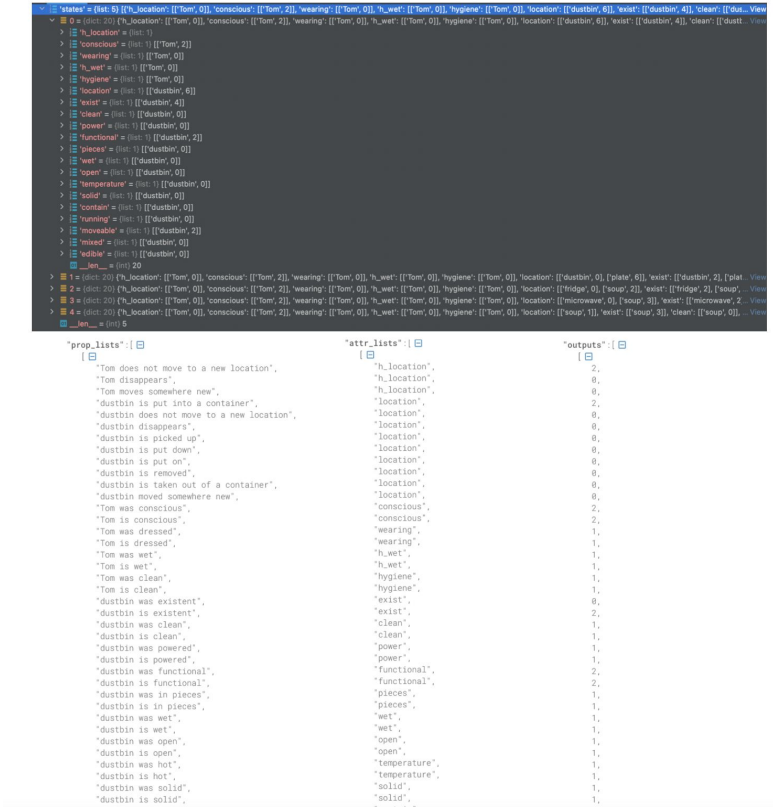
TRIP instance for example



The same instance after conversion


```
{
  "story_id":0,
  "example_id":"0-C0",
  "idx":0,
  "texts":[
    "Tom bought a new dustbin for the kitchen.",
    "Tom threw a broken plate in the dustbin.",
    "Tom got some soup from the fridge.",
    "Tom put the soup in the microwave.",
    "Tom ate the cold soup."
  ],
  "breakpoint":4,
  "confl_sents":[
    3
  ],
  "confl_pairs":[
    [
      3,
      4
    ]
  ],
  "prop_lists":[
    [ ],
    [ ],
    [ ],
    [ ],
    [ ]
  ],
  "attr_lists":[
    [ ],
    [ ],
    [ ],
    [ ],
    [ ]
  ],
  "outputs":[
    [ ],
    [ ],
    [ ],
    [ ],
    [ ]
  ],
  "guid":"0-C0-d862c2e3-6b0a-42f7-827c-67ebc8d44df7"
}
```

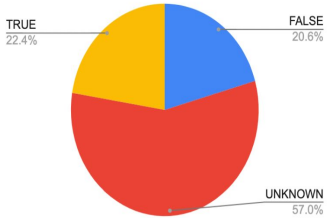
Data format comparison - (states of the first sentence)



Experiment: Applying Breakpoint Model to TRIP

Hyperparameters

Training data distribution(implausible stories):
0=unknown, 1=false, 2=true



- **Unknown under sampling ratio:**
  - 35% gives us balanced data in training.
  - We also checked 25%, 17.5% and 10%.
  - **In verifiability evaluation of TRIP, they take non-default values predictions**.
  - Most of the attribute default values are 0=unknown.

```
att_default_values = {'h_location': 0, 'conscious': 2, 'wearing': 0, 'h_wet': 0, 'hygiene': 0,
                      'location': 0, 'exist': 2, 'clean': 0, 'power': 0, 'functional': 2,
                      'pieces': 0, 'wet': 0, 'open': 0, 'temperature': 0, 'solid': 0, 'contain': 0,
                      'running': 0, 'moveable': 2, 'mixed': 0, 'edible': 0}
```

- **#Epochs:**
  - We considered runs with 25 and 30 epochs.

- **Other hyperparameters:**
  - We didn't tune other hyperparameters. We could tune also **learning rate** and **batch size**.
  - Our experiment have been done with batch size of 1, gradient accumulation steps of 8 and learning rate of 0.00005.

Metrics for evaluation

- **Metrics for general data evaluation(all implausible stories, all sentences, all predictions):**
  - micro F1 score(=accuracy)
  - macro F1 score(average F1 scores of the classes)
  - precision, recall and F1 score per class

- **Metrics for verifiability:**
  - %verifiable stories / consistent stories
  - Macro F1 score per attribute in consistent stories for sentences of precondition, effect and in total.

Considerations for evaluation

- Comparison of the location attributes

Sentence type	**sentence**	**prop**	**Roberta-Large prediction**	**label**	**Is Roberta-Large correct**	**Breakpoint Model Prediction**	**Is Breakpoint Model correct**
effect	Tom spilled the glass of milk on the floor.	milk **is put into a container**	is put into a container	is put down	FALSE	FALSE	TRUE

  - For every location proposition, our model needs only to predict False/ True for every independent propositions, however in TRIP they predict the specific location (classification problem)
    - **Easier** for our model if only needs to predict TRUE for correct proposition
    - **Harder** for our model if also needs to predict all false propositions correctly.

```
'location': {0: 'does not move to a new location', 1: 'disappears', 2: 'is picked up', 3: 'is put down',
             4: 'is put on', 5: 'is removed', 6: 'is put into a container',
             7: 'is taken out of a container', 8: 'moved somewhere new'},
```

- Use 10 different unknown sampling seeds for every sampling ratio

Metrics for evaluation

- **Metrics for general data evaluation(all implausible stories, all sentences, all predictions):**
  - micro F1 score(=accuracy)
  - macro F1 score(average F1 scores of the classes)
  - precision, recall and F1 score per class

- **Metrics for verifiability:**
  - %verifiable stories / consistent stories
  - Macro F1 score per attribute in consistent stories for sentences of precondition, effect and in total.

- **Considerations for the evaluation:**
  - Comparison of the location attributes
    - Other possible comparison method for location attributes
  - Use 10 different seeds for every sampling ratio

Breakpoint Model evaluation

Breakpoint Model Vs. TRIP best run(Roberta-Large)

TRIP best run(Roberta-Large) results:

- **TRIP results on the three proposed tasks:**

dataset	verifiable pred.	consistent pred.	inconsistent pred.	wrong story pred.	total implausible stories
dev	34(10.6%)	72(22.4%)	165(51.2%)	85(26.4%)	322
test	32(9.1%)	67(19.1%)	189(53.8%)	95(27.1%)	351

- **TRIP results on verifiability:**

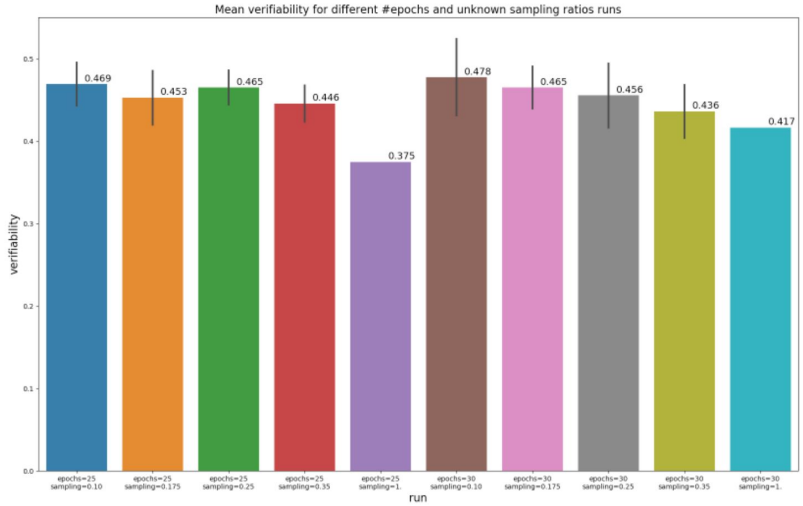
dataset	total propositions	avg. propositions per story	verifiable pred.	consistent pred.	verifiable pred. / consistent pred.
dev	191	2.938	34	72	**47.2%**
test	142	2.491	32	67	**47.8%**

- **TRIP results on verifiability(assuming all location attributes are correct)**

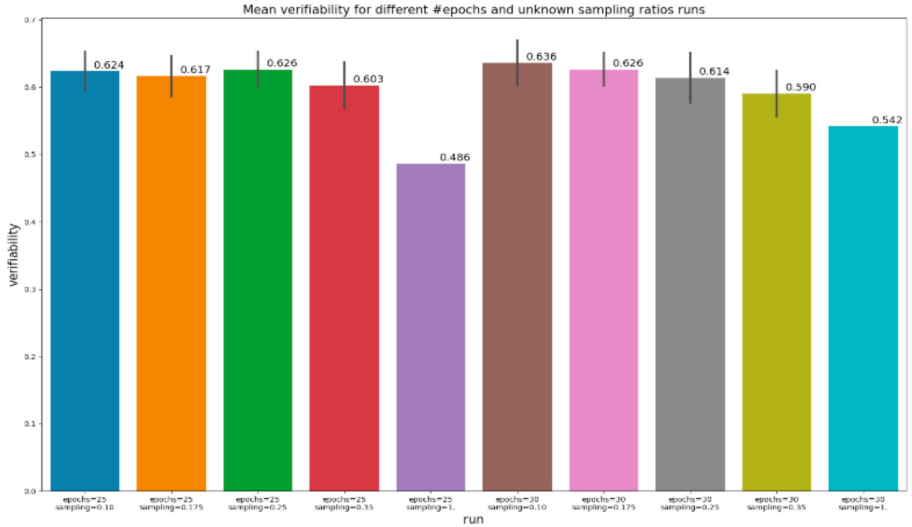
dataset	total propositions	avg. propositions per story	verifiable pred.	consistent pred.	verifiable pred. / consistent pred.
dev	191	2.938	48	72	**66.7%**
test	142	2.491	45	67	**67.2%**

Breakpoint Model verifiability on dev set

Including location attributes predictions as is



Given that location attributes predictions are correct

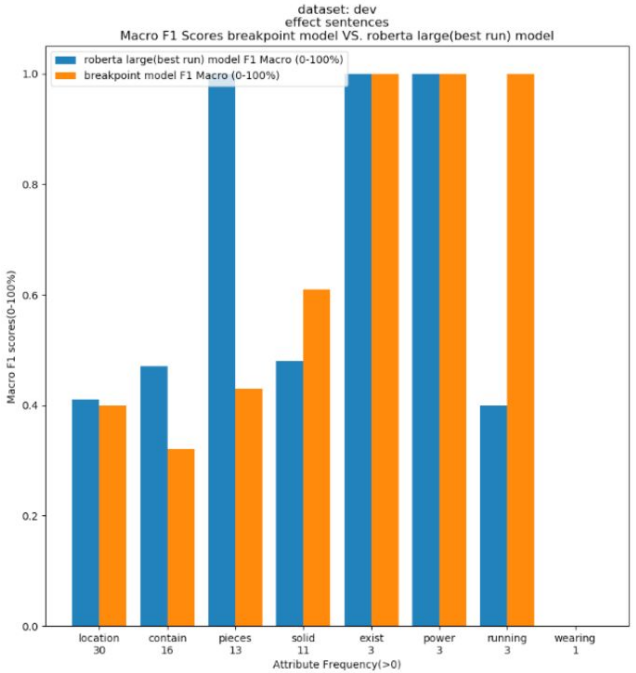


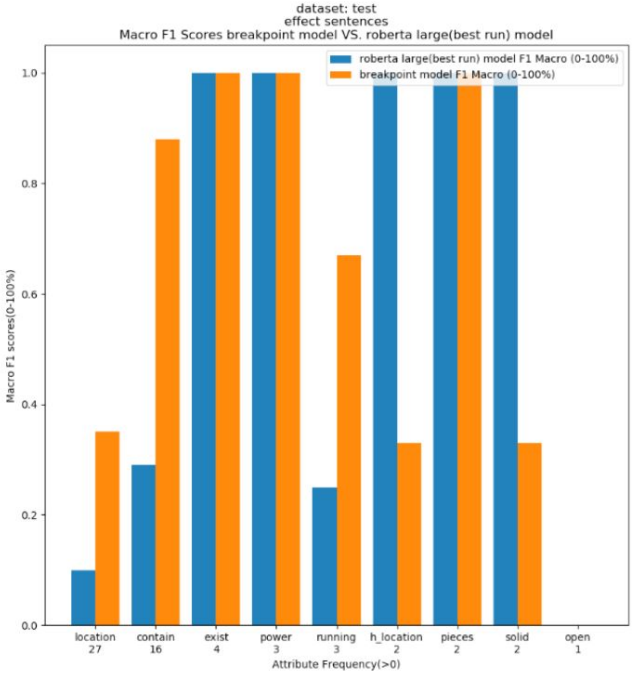
neutralize location attributes	best verifiability(**ours**)	mean verifiability(**ours**)	verifiability(**Roberta-Large**)
no	55.6%	47.8%	47.2%
yes	66.7%	63.6%	66.7%

Breakpoint Model verifiability Vs. Roberta-Large on test set

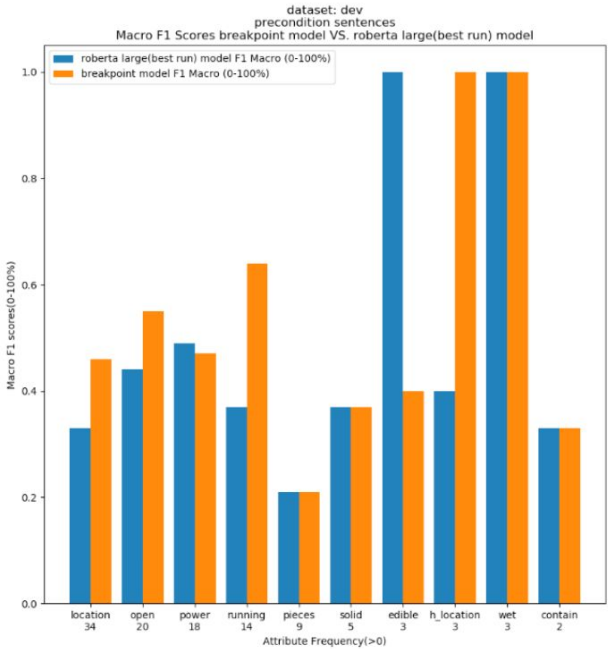
neutralize location attributes	verifiability(**ours**)	verifiability(**Roberta-Large**)	diff
no	$(31/67) = \mathbf{46.3\%}$	$(32/67) = 47.8\%$	-1.5%
yes	$(46/67) = \mathbf{68.7\%}$	$(45/67) = 67.2\%$	$+1.5\%$

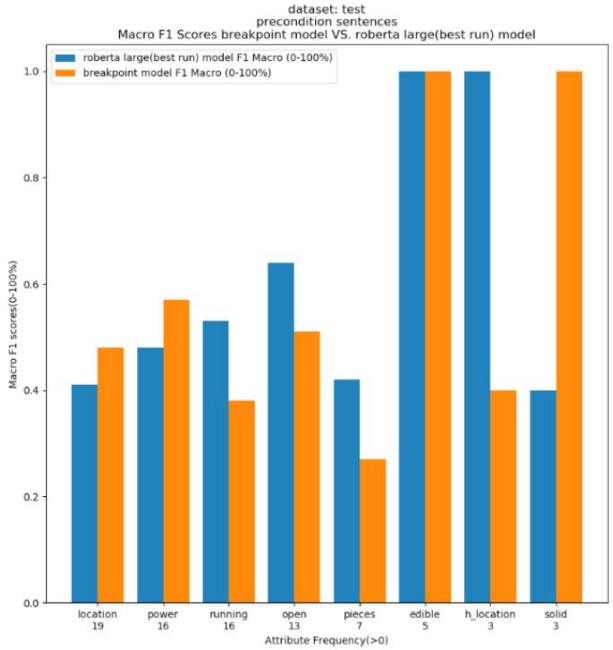
Breakpoint Model Vs. Roberta-Large Macro F1 scores on attributes



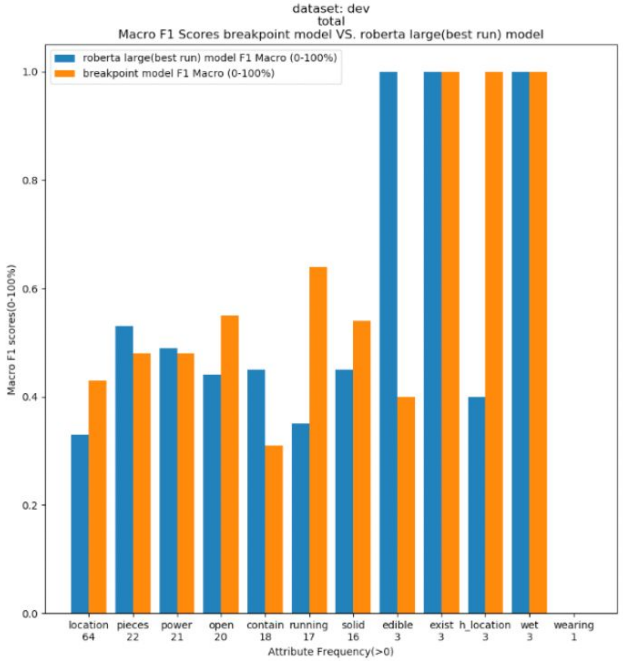


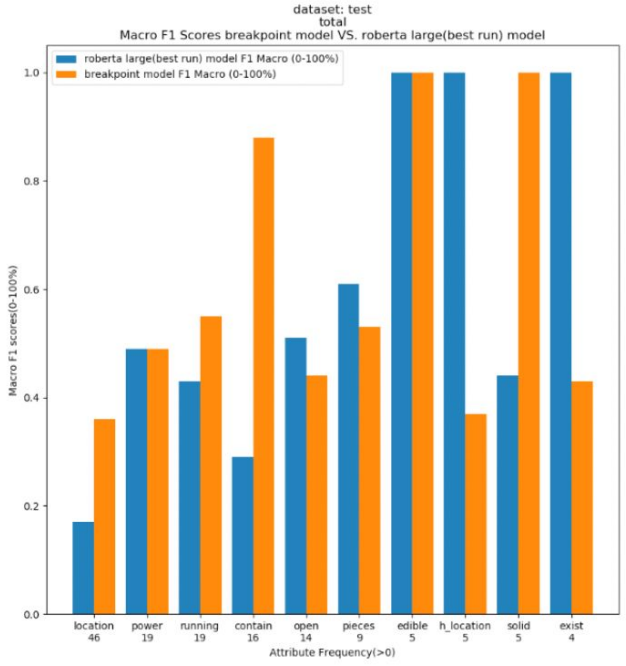
Breakpoint Model Vs. Roberta-Large Macro F1 scores on attributes





Breakpoint Model Vs. Roberta-Large Macro F1 scores on attributes





Breakpoint Model General evaluation

- Here we looked at all of the implausible stories, for each story, in all sentences, for each sentence, in all of the predictions of the different attributes and entities.

- **Dev set:**

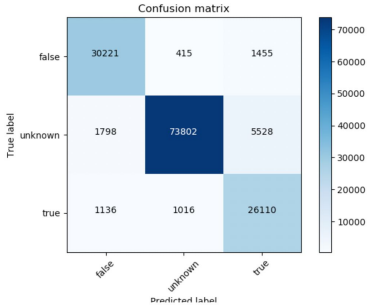
#epochs	*unknown* random sampling	seed	#{*false*, *unknown*, *true*} in training set	Micro F1(=accuracy)	Macro F1
30	10%	12348	{63366, 16087, 58175}	93%	92%
30	100%	–	{63366, 160864, 58175}	95%	93%

- **Test set:**

#epochs	Micro F1(=accuracy)	Macro F1
30	92%	91%

Breakpoint Model General evaluation

- **Confusion Matrix:**



The model predicts true instead of unknown in 5528 out of 5528+1455+26110=33593 true predictions

- **Precision, recall and f1 score per class:**

	precision	recall	f1 score	# instances
false	91%	94%	93%	32091
unknown	98%	91%	94%	81128
true	79%	92%	85%	28262
accuracy			92%	141481

Any Questions ?

Code in Github:

https://github.com/orensul/situation_modeling/tree/run_trip_no_pre_post/trip

Thanks for listening