# Improvement on Baseline System for Tiered Reasoning for Intuitive Physics

**Wenxin He**
University of Michigan
wenxinhe@umich.edu

**Ruge Xu**
University of Michigan
rugexu@umich.edu

**Yixin Shi**
University of Michigan
esing@umich.edu

## Abstract

The Storks paper introduces TRIP, a rich-annotated commonsense reasoning dataset that enables a multi-tiered evaluation of machines' reasoning processes. It also proposed a machine-learning reasoning model that is trained on this dataset. The low performance on the system's verifiability calls for better language-understanding models. We tried to improve the baseline system performance in three ways: automatic data augmentation, loss function change, and adding a new module. We evaluate the improved system using three metrics: accuracy, consistency, and verifiability. Results show that the methods have successfully improved the performance in some way.

## 1 Introduction

Commonsense reasoning is one of the heated topics in natural language understanding. Recent research activities have achieved impressive performance in challenging language understanding tasks. More and more large amounts of large-scale benchmark datasets and language models have been developed. Some of them even achieve extraordinaty performance that surpassed human performance in challenging language understanding tasks. However, evaluations only based on end-task performance don't necessarily mean that machines have the true ability in language understanding and reasoning. To deal with this problem, the dataset Tiered Reasoning for Intuitive Physics (TRIP) is proposed (Storks et al., 2021), which enables a multi-tiered evaluation of machines' reasoning processes. Though current several baseline systems can achieve high accuracy (78%) on TRIP, only a small portion (11%) of prediction is supported by proper evidence. The main task of this project is to modify the baseline system and improve the system performance, especially consistency and verifiability.

## 2 Related Work

Paper(Storks et al., 2021) proposes Tiered Reasoning for Intuitive Physics (TRIP) dataset to enable multi-tiered evaluation of machines' reasoning process. Intuitive physics problem is always thought to be especially challenging for machines because physical commonsense is considered obvious to most humans, and suffers from reporting bias ((Forbes and Choi, 2017)). TRIP dataset aims at evaluating not only the end-task performance but also the process of reasoning by story plausibility classification, a common proxy task for commonsense reasoning problems. In the paper, three pre-trained language models are considered: BERT(Jacob Devlin and Toutanova, 2018), RoBERT$_A$(Yinhan Liu and Stoyanov, 2019), and D$_E$BERT$_A$(Pengcheng He and Chen, 2021). The TRIP dataset utilizes rich annotation of physical states, and the physical state attributes were collected from attribute spaces proposed in the paper(Qiaozi Gao and Chai, 2016) and (Antoine Bosselut and Choi, 2018). For the contextual embedding part of the module implementations, the input is formed following an entity-first input formulation proposed in the paper(Gupta and Durrett, 2019). Similar to TRIP's physical state classification, existing dataset ProPara(Mishra et al., 2018), tracks the location and existence of entities in each sentence, and the paper also introduces models for the state prediction.

## 3 Dataset

The Tiered Reasoning for Intuitive Physics (TRIP) is a benchmark for physical common-sense reasoning that provides a trace of reasoning for the ultimate task of plausibility prediction. The dataset consists of human-authored stories, each describing a sequence of physical actions. In a given pair of highly similar stories, most of them are identical and plausible sentences, but one of the stories contains an unplausible sentence, thus making the story physically false. The corresponding task is to distinguish which story is plausible (Storks et al., 2021).

The plausible stories were obtained from the Amazon Mechanical Turk. Each participating worker is assigned a task to independently write a new sentence to replace one of the sentences in the original plausible story, making the replaced new story no longer realistic in the physical world. Also, to ensure quality, workers flagged stories that were incoherent or did not describe physical actions, and these stories were eliminated after manual verification.

TRIP includes annotations for each story that contain multiple layers of reasoning beyond the final task. Thus, a hierarchical assessment of the machine's coherence becomes feasible. The system will assess (1) the ability to discriminate plausible stories, (2) the ability to identify conflicting sentences in implausible stories, and (3) the ability to identify the underlying physical states in sentences.

## 4 Approaches

### 4.1 Automatic Data Augmentation

In the original paper, the author points out that in low-level tasks such as the physical states classification and conflict detection, the training loss decreases slowly. Other evidence such as the model beginning to overfit also suggests a need for a larger training data set. Thus, we want to automatically generate more training data on the human-annotated data set. We proposed the following two main methods.

#### 4.1.1 Adding Repetition

This method is used to generate new implausible stories from the existing plausible stories. In the original plausible story, we focus on any sentences that contain an entity that has a non-idempotent physical state. A non-idempotent physical state cannot be repeated, so duplicating this sentence will make this story implausible. For example, in the sentence "John opened the fridge and got out the pizza.", the "location" state of "pizza" is of the label "taken out of the container", which cannot be done twice. Thus, we can replace another sentence in the story with this sentence and make a new implausible story. Other non-idempotent physical state examples are:

- "conscious": true->false
- "wearing": false->true
- "hygiene": true->false
- "power": true->false
- "functional": true->false
- "pieces": true->false
- "contain": false->true
- "mixed": false->true
- "edible": false->true

In the TRIP data set, we perform this data augmentation on the cloze-type subset of the training set. In a cloze-type story pair, we retrieve the original plausible story, detect the non-idempotent sentence, and replace another sentence in the story with this sentence, and append this new cloze pair to the training set.

#### 4.1.2 Reordering Sentences

This method is also used to generate new implausible stories from the existing plausible stories. In the original plausible story, we focus on a sentence pair that has an ordering requirement, and we switch the two sentences to make the story implausible. For example, entity E in sentence A has a physical state PS of the label "true->false", and the same entity E in sentence B has the physical state PS of the label "false->false". In the original plausible story, sentence A comes before sentence B, and the logical relation holds. However, if we switch A and B, the story will become implausible because the physical state PS cannot change from true to false if its previous state is false. Example sentence pairs are:

- "open": true->false, then false->false
  false->true, then true->true
- "wet": true->false, then false->false
- "h_wet": true->false, then false->false
- "conscious": true->false, then false->false
  false->true, then true->true
- "clean": true->false, then false->false
- "power": true->false, then false->false
  false->true, then true->true

- "solid": true->false, then false->false
- "edible": true->false, then false->false
- "functional": true->false, then false->false

  false->true, then true->true

In the TRIP data set, we perform this data augmentation on the order-type subset of the training set. In an order-type story pair, we retrieve the plausible-labeled story, scan over the entities and their corresponding physical states, identify the re-orderable sentence pairs and switch them. Then we append the new order-type story pair to the training set.

We have noticed that the order-type subset already contains this consideration, as there are several stories based on the same story ID, and the difference among them is the ordering of certain sentence pairs. However, this annotation is done by humans and doesn't cover all possible reordering of sentences. Our algorithm can detect all permutations of reorderable sentence pairs, and compare them to existing stories in the order-type subset to avoid duplications. In this way, our algorithm makes the most of one plausible story, and this data augmentation can also be applied to future human annotation processes in generating the order-type subset.

We believe that by adding these automatically generated data to the training set, the model can learn the implied relationship and thus delivers better performance.

## 4.2 Improvement on the Loss Function

The previous paper uses four loss functions for training the tiered baseline system, including $L_p$ for precondition classification, $L_f$ for effect classification, $L_c$ for conflicting sentence detection, and $L_s$ for story choice classification. Based on the four loss functions, it trains the baseline model's parameters through gradient descent on the overall loss $L$, which is a linear combination of the four loss functions:

$$L = \lambda_p L_p + \lambda_f L_f + \lambda_c L_c + \lambda_s L_s$$

Results in the paper show that using all losses gives the best accuracy while omitting story choice loss gives the best consistency and verifiability.

### 4.2.1 Change the loss weights

Limited loss weight combination has been tested in the paper. Thus, we tried more combinations in this project. To get a comprehensive view, we plan tested the new combiantions both on origianl dataset and augmented dataset. Previous paper has shown that omitting story choice loss ($\lambda_s$) gives the best performance. Thus, we kept $\lambda_s$ = 0. The following is the set of weight combinations that we have experimented on:

| **No.** | $\lambda_p$ | $\lambda_f$ | $\lambda_c$ | $\lambda_s$ |
|---|---|---|---|---|
| 1 | 0.4 | 0.4 | 0.2 | 0.0 |
| 2 | 0.3 | 0.3 | 0.4 | 0.0 |
| 3 | 0.2 | 0.2 | 0.6 | 0.0 |

Table 1: Loss Weight Combination

### 4.2.2 Step-by-step weight

We noticed that for the baseline model, the best accuracy and the best verifiability fail to exist in the same model. We guess that this may result from the model's failure to grab enough information about each step. The selection of loss functions is an important factor influencing the information a model gets. Thus, the idea we come up with to deal with the problem is to use a step-by-step overall loss. That is, instead of training the model's parameters using a linear sum of the four loss functions at a time, we train the parameters based on different loss functions step by step. We first train the model using the loss for precondition classification ($L_p$), then further trained the model based on the previously trained model using the loss function for effect classification ($L_f$). Repeat the same procedure for loss for conflicting sentence detection ($L_c$) and story choice classification ($L_s$).

## 4.3 Improvement on the Conflict Detector

Currently, although the baseline system is relatively accurate in determining whether sentences are plausible, it is not very consistent in predicting which pairs of sentences conflict. We hope to improve the conflict detector module of the baseline system as a way to raise the consistency of the predictions.

As can be seen from the previous experiments, when the physical state logit from the precondition and effect classifier is added to the input of the conflict detector, the improvement in accuracy, consistency, and verifiability is not significant, and is even lower than the performance when only the contextual embedding of sentence-entity pairs is used as input. The performance of the model is horribly low when using only physical states as input (Storks et al., 2021).

We hope that a module can be added to preprocess the physical state to extract more information for the purpose of improving the performance of the baseline system, as shown in Figure 1.

### 4.3.1 Structure of the Physical State Processing Module

Adding a module means that before the physical states of an object enter the conflict detector module, they enter another module that forces the model to learn information about the physical states of the object. The more specific structure of the new physical state processing module is shown in Figure 2. The input of the module is the physical state of the object and the output is the physical state of the object after being processed.

The object's physical state is a simple concatenation of the precondition states of the object and the effect states of the object. The shape of this variable is:

$$(B \times N \times E \times S, M)$$

where $B$ is the batch size, $N$ is the number of stories, $E$ is the number of entities, $S$ is the number of sentences, and $M$ is the state size. In order to keep the model simple and clear and to change the original model as little as possible, we wanted the new module to be consistent in the shape of its inputs and outputs.

### 4.3.2 Module Selection

To ensure that the inputs and outputs have the same shape and that no additional variables are added, we will consider the following modules.

**Linear Module.** This module applies a linear transformation to the input data. The conversion formula is $y = xA^T + b$. The shapes of its input and output are:

$$Input: \ (*, H_{in})$$

$$Output: \ (*, H_{out})$$

where $*$ means any number of dimensions, $H_{in}$ is the input dimension and $H_{out}$ is the output dimension. Here, $*$ refers to $B \times N \times E \times S$ and $H_{in}$ and $H_{out}$ are set to $M$.

**RNN Module.** This module applies a multi-layer Elman RNN with ReLU non-linearity to an input sequence. It requires the initial hidden state in addition to the input sequence. The shapes of its input, initial hidden state, and output are:

$$Input: \ (B, L, H_{in})$$
$$h_0: \ (num\_layers, H_{out})$$
$$Output: \ (B, L, H_{out})$$

where $L$ is the sequence length. Here, the number of layers is set to 1, $L$ is set to $N \times E \times S$, and $H_{in}$ and $H_{out}$ are set to $M$.

**LSTM Module.** This module applies a multi-layer long short-term memory (LSTM) RNN to an input sequence. It requires the initial hidden state and the initial cell state in addition to the input sequence. The shapes of its input, initial hidden state, initial cell state and output are:

$$Input: \ (B, L, H_{in})$$
$$h_0: \ (num\_layers, H_{out})$$
$$c_0: \ (num\_layers, H_{cell})$$
$$Output: \ (B, L, H_{out})$$

where $H_{cell}$ means the hidden size. Here, the number of layers is set to 1, $L$ is set to $N \times E \times S$, and $H_{in}$, $H_{cell}$ and $H_{out}$ are set to $M$.

**GRU Module.** This module applies a multi-layer gated recurrent unit (GRU) RNN to an input sequence. It requires exactly the same inputs and outputs and the same shape as RNN, which is not repeated here.

**Tranformer Encoder.** This module is made up of a self-attention and feed-forward network, based on the paper Vaswani et al. (2017). The shapes of its input and output are:

$$Input: \ (*, d\_model)$$

$$Output: \ (*, d\_model)$$

where $d\_model$ is the number of expected features in the input. Here, $d\_model$ is set to $M$.

## 5 Evaluation

The TRIP dataset evaluates the performance of the model from three perspectives in order to test the model's coherent reasoning ability.

**Accuracy.** This metric tests the ability of the model to identify the correct plausible story, using the percentage of correct instances as the test criterion.

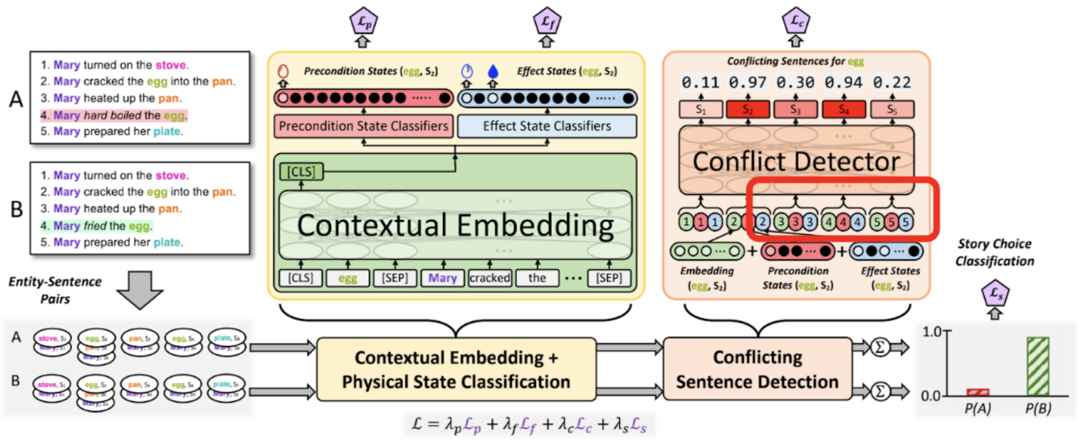
A
1. Mary turned on the stove.
2. Mary cracked the egg into the pan.
3. Mary heated up the pan.
4. Mary hard boiled the egg.
5. Mary prepared her plate.
B
1. Mary turned on the stove.
2. Mary cracked the egg into the pan.
3. Mary heated up the pan.
4. Mary fried the egg.
5. Mary prepared her plate.
Entity-Sentence Pairs
Precondition States (egg, S2)
Effect States (egg, S2)
Precondition State Classifiers
Effect State Classifiers
[CLS]
Contextual Embedding
[CLS] egg [SEP] Mary cracked the ... [SEP]
Conflicting Sentences for egg
0.11 0.97 0.30 0.94 0.22
Conflict Detector
Embedding (egg, S2)
Precondition States (egg, S2)
Effect States (egg, S2)
Story Choice Classification
Contextual Embedding + Physical State Classification
Conflicting Sentence Detection
1.0
0.0
P(A)
P(B)
$\mathcal{L} = \lambda_p\mathcal{L}_p + \lambda_f\mathcal{L}_f + \lambda_c\mathcal{L}_c + \lambda_s\mathcal{L}_s$

Figure 1: Proposed tiered reasoning system in Storks et al. (2021). The red box represents where we want to add a new module.

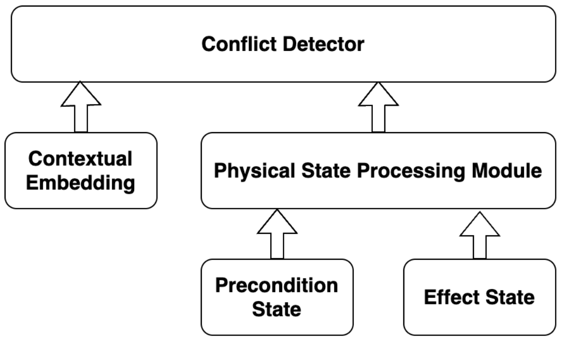

Figure 2: Detailed structure of the new physical state processing module.

**Consistency.** This metric tests the ability of the model to identify conflicting sentence pairs in an implausible story based on the identification of the correct plausible story. It uses the proportion of correct sentence pairs as the test criterion.

**Verifiability.** This metric tests the ability of the model to identify the physical state of the object causing the conflict based on identifying conflicting sentence pairs in the implausible story and the correct plausible story. It uses the proportion of correct conflicting physical states as the test criterion.

In the actual test, we will see that the indicators of the model show the following relationship: $accuracy \geq consistency \geq verifiability$. We expect a model whose judgments pass all the test steps, i.e. whose performance exhibits the characteristics that $accuracy \approx consistency \approx verifiability$.

Currently, the baseline model in Storks et al. (2021) has very low verifiability although it achieves a high value in accuracy. In the next experiments, we mainly focus on the verifiability of the improved model as well and compare it with the verifiability of the baseline.

## 6 Results

### 6.1 Reproduction of Baseline

First, we reproduced the results from Storks et al. (2021). Figure 3 shows that the highest accuracy of this system is about 77.02% and the highest verifiability is 10.25%, which is very low. Not only that, we can find that the models all reach the best performance at epoch 5, after which neither accuracy nor verifiability can be improved, but only slowly decline.

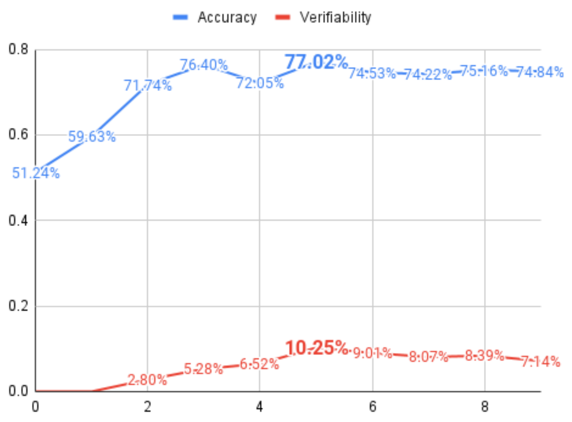

Figure 3: Accuracy (blue) and verifiability (red) for best tiered RoBERTa system on the validation set trained on TRIP for 10 epochs.

Our purpose is to improve a model whose verifiability can be higher than 10.25%, while accuracy does not have a significant decrease.

For our experiments, we used the learning rate(1e-5) and batch size(1) same as the baseline model in the paper.

## 6.2 Automatic Data Augmentation

For the method mentioned in 4.1.1, we select three physical states: "location", "conscious", and "wearing". From a total of 799 cloze-type stories in the training set, we generate 49 new implausible stories.

For the method mentioned in 4.1.2, we select nine physical states: "open", "wet", "conscious", "h_wet", "clean", "power", "solid", "edible", and "function". From a total of 2330 order-type stories in the training set, we generate 7 new implausible stories(the duplicated ones are removed and thus not counted).

## 6.3 Improvement on the Loss Function

The following table shows the performance of the system with different loss weight training on the dataset TRIP.

| Loss Weight | Accuracy | Verifiability | Consistency |
|---|---|---|---|
| (.4,.4,.2,0) | 77.0 | 10.2 | 28 |
| (.3,.3,.4,0) | **79.2** | 7.1 | **32.3** |
| (.3,.3,.4,0) | 77.8 | **10.8** | **35.5** |
| (.2,.2,.6,0) | **79.5** | 8.5 | **33.0** |
| (.2,.2,.6,0) | 77.3 | **10.4** | **35.8** |

Table 2: Metrics for the different loss weight on the different training set of TRIP. Black loss weight is trained on original TRIP dataset while brown loss weight is trained on augmented TRIP dataset. The first row represents the original baseline.

From the table above, we see that the model with $\lambda_p = 0.3$, $\lambda_f = 0.3$, $\lambda_c = 0.4$, $\lambda_p = 0.0$, that using the augmented data gives the best performance. In detail, comparing row 1 and row 2, row 1 and row 4, we see that increasing the weight of conflict detector loss improves consistency. Comparing row 2 and row 3, row 4 and row 5, we find that data augmentation improves the verifiability and consistency.

The following table shows the performance of the system with a step-by-step loss function for different epoches.

Table 3 shows that though using the step-by-step loss function slightly improves the end-task accuracy (baseline accuracy 77%), this method does not improve the verifiability.

| Epoch | Accuracy | F1 | Verifiability |
|---|---|---|---|
| 3 | 78.3 | 78.3 | 0.0 |
| 5 | 75.6 | 75.5 | 0.0 |
| 7 | 74.7 | 74.5 | 0.0 |

Table 3: Metrics for step-by-step loss weight on the training set of TRIP.

## 6.4 Improvement on the Conflict Detector

We added different modules to process the physical state of the object separately and checked the performance of the new model. The trends of accuracy, f1 score, and verifiability of the original model and the new models over the course of training 10 epochs are shown in Figure 4. As can be seen, regardless of the model, their accuracy and f1 scores are very close and it is difficult to continue improving after about 3-4 epochs. Among them, the system with the addition of the LSTM module can reach the highest accuracy, up to 80%, while the other models can only reach a maximum of 79% accuracy.

For verifiability, the time to reach the maximum verifiability varies from system to system, but all reach the maximum within 10 epochs. After reaching the maximum point, the verifiability of the system starts to fluctuate erratically or decrease slowly. This means that continued training will not improve the verifiability of the system. Among them, the system with the addition of the LSTM module and the addition of the RNN module can achieve the highest verifiability.

The best performance achieved in 10 epochs for each system is presented in Table 4. Some systems have their verifiability beyond the verifiability of the baseline after adding modules, including systems that add a linear module, systems that add an LSTM module, and systems that add an RNN module. Among them, the system with the addition of the linear system has only a small improvement in verifiability, while the system with the addition of the LSTM module and the system with the addition of the RNN module were able to achieve the highest verifiability, i.e., 11.2%. Other systems are unable to surpass the verifiability of the baseline, including systems that add the GRU module and systems that add the Transformer Encoder module. This may be due to the fact that these two modules are too complex to be adequately trained.

When we focus on the accuracy of the system, we find that only the system with the addition of

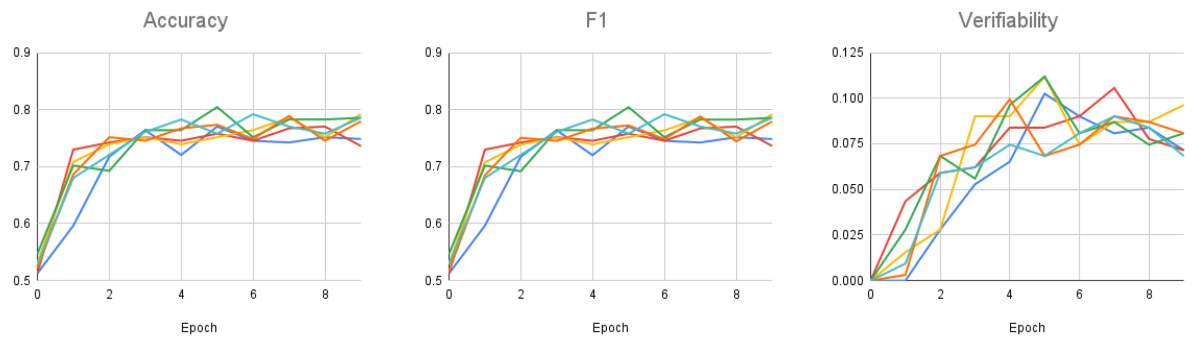

Figure 4: Accuracy (left), f1 score (middle), and verifiability (right) for the models trained on TRIP for 10 epochs. The models are the original baseline (blue), the model with a linear module (red), the model with an RNN module (yellow), the model with an LSTM module (green), the model with a GRU module (orange), and the model with a Transformer Encoder (aqua).

the LSTM module can improve verifiability while the accuracy is not only not lower than the baseline one, but also has a slight improvement to 80%. The system with the addition of the RNN module can also get 11.2% verifiability, but at the same time accuracy is lower than that of baseline. Therefore, we consider the system with the addition of the LSTM module to be optimal.

| Model | Accuracy | F1 | Verifiability |
|---|---|---|---|
| Baseline | 77.0 | 77.0 | 10.2 |
| Linear | 76.7 | 76.7 | 10.6 |
| RNN | 75.2 | 75.1 | **11.2** |
| LSTM | **80.4** | **80.4** | **11.2** |
| GRU | 76.7 | 76.7 | 9.9 |
| Transformer | 77.0 | 77.0 | 9.0 |

Table 4: Metrics for the best systems on the training set of TRIP. Compared to the original baseline.

## 7 Discussion

### 7.1 Automatic Data Augmentation

For the two augmentation methods, we didn't apply the augmentation on all eligible sentences. This is because we don't want the model to exploit these trivial features and learn bias, which will result in overfitting. As the newly-generated data is small in size, we propose that more types of data augmentation can be designed, such as adding negations, passive voice transformation, tense conversion, and paraphrasing. While constructing the algorithms or transformations can be time-consuming, there exist state-of-the-art methods for automating the data augmentation process. TANDA(Ratner et al., 2017) proposes a framework to learn the augmentation by modeling data augmentation as transformation functions sequences and has achieved high efficacy on text dataset. Future work may consider this model to enlarge the dataset in a time-saving fashion.

Regarding the physical states annotations in the TRIP dataset, more detail should be provided. We propose that objects of a verb can be included in the annotation. For example, the 'location' state can only be labeled as 'put on', 'put into container', 'taken out of contained', etc. In this case, the system learns nothing from the object or the container such as a fridge or a bowl. This information should be fed into the system since the system makes its decision largely based on the precondition and effects of the physical states. A separate machine learning model can be applied to learn this relationship and enhance the annotation. What's more, a total of only 20 physical state attributes are selected. We suggest that more attributes and labels can be added such as human moods, humans relationship, ability, etc.

As TRIP is a relatively small dataset, we can also fine-tune the pre-trained model based on other NLP commonsense reasoning dataset. ROCStories(Mostafazadeh et al., 2016) is a corpus of 50k commonsense stories each consisting of 5 sentences and provides a hypothesis or implication for each story. ROCStories is very similar to TRIP in nature. Another commonsense dataset, ATOMIC(Sap et al., 2018), is a collection of 900k textual variables of if-then relations, and can help models learn the inferential relationships in events. By fine-tuning the pre-trained model, we hope the system can capture temporal and causal relationships between common daily events from other mature and large datasets, thus making up for the small size of TRIP.

### 7.2 Experiment on the Loss Weight with Augmented Dataset

From Table 2, we find increasing the weight for conflicting sentence detection loss gives an improvement in both accuracy and consistency. If training the augmented data, the performance in verifiability and consistency further improves. We think the improvement may result from the definition of consistency and verifiability. Since both consistency measures how many stories that are identified correctly with the exact correct conflict pairs and verifiability measures how many stories give both correct conflict pairs and physical states, the conflict detector plays a major role in evaluation. Thus, it is reasonable that increasing the weight for conflict detection loss improves consistency obviously.

Moreover, by comparing row 2 and row 3, row 4 and row 5 in Table 2, we find augmented dataset improves verifiability and consistency obviously. This shows, by adding repetition and reordering sentences, the model hopefully learned the implied information as stated in section 4.1.1 and 4.1.2.

We also tried step-by-step loss on all loss functions for the model. Table 3 shows though the model still gives high (even higher) end-task accuracy, unfortunately, there is no improvement in verifiability. We think the reason may be that training the model step by step will weigh more for later loss function. For this system, story choice classification weighs the most. Thus, the classification reaches a higher accuracy. However, step-by-step loss pays less attention to the first few losses like loss for physical state and conflict detector, making the performance of consistency and verifiability not good.

### 7.3 Experiment on the Conflict Detector

When we add new modules, we keep all the outputs of the recurrent layers (including RNN, LSTM, and GRU) in order to ensure that the input and output shapes are consistent. In fact, for these recurrent layers, we usually only need the output of the last hidden state, because the output of the first few hidden states has not gone through many cycles and does not contain much information. There is no way to guarantee whether this information serves as an enhancement to the overall system or as redundant information that disrupts the system's judgment.

Also, the output of recurrent layers should go through a fully connected layer to change its output to the desired output shape, while we directly set the output of recurrent layers to the desired shape. We made the change for the sake of the simplicity of the system, but we do not know if the change will have any additional impact on the interpretability of the output.

Finally, the initial state setting of the hidden state and the cell state is also an issue. Currently, we set them to 0. In fact, there is the option of setting them to random numbers, which may affect the stability of the system, but may also improve its performance of the system further. We have not yet made such an attempt.

Therefore, if available, we can try to (1) keep only the output of the last hidden state of recurrent layers, (2) modify the output of recurrent layers to our desired shape by a fully connected layer, and (3) compare the effect of different initial state settings of hidden state and cell state on the performance of the system.

## 8 Conclusion

In this project, we not only practiced our ability in reading and reproducing a paper, but also tried to propose our new ideas to improve the model. We have all first understood the target paper thoroughly and successfully reproduced the result, just like what we have learned in homework 4. Furthermore, we consider improving the baseline model from three different aspects: dataset augmentation, loss function and system architecture. We then evaluated our revised models using different metrics: accuracy, consistency and verifiability. Compared with the results in the paper, most of our methods have successfully improved the performance while some haven't. We thoroughly analysed all the methods with their corresponding results, trying to explain the result. In the project, we distributed the work properly as well as cooperated with each other. Through trials and fails, we have earned a deeper understanding of NLP problems and learned to solve them at the same time.

## Acknowledgements

This work was a course project for EECS 595 Natural Language Processing at the University of Michigan. We sincerely thank Professor Joyce Chai and the Graduate Student Instructors Peter Yu, Martin Ma, and Shane Storks for their help in our work and for their helpful comments and suggestions on our project.

## References

Ari Holtzman Corin Ennis Dieter Fox Antoine Bosselut, Omer Levy and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *Proceedings of the 6th International Conference on Learning Representations(ICLR 2018)*.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Aditya Gupta and Greg Durrett. 2019. Effective use of transformer networks for entity tracking. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Linguistics*.

Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceesings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies(NAACL HLT 2019)*.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories.

Jianfeng Gao Pengcheng He, Xiaodong Liu and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentagled attention. *arXiv:2006.03564*.

Shaohua Yang Qiaozi Gao, Malcolm Doering and Joyce Chai. 2016. Physical causality of action verbs in grounded language understanding. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics(ACL 2016)*.

Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2018. Atomic: An atlas of machine commonsense for if-then reasoning.

Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. 2021. Tiered reasoning for intuitive physics: Toward verifiable commonsense language understanding. *arXiv preprint arXiv:2109.04947*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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