Pretrained Commonsense Language Models and Memory Networks for TRIP

Muhammad Khalifa
khalifam@umich.edu

James Tavernor
tavernor@umich.edu

Abstract

In this project, we investigate whether it is possible to improve the underlying reasoning of the Tiered Reasoning for Intuitive Physics (TRIP) baseline model. We look at the consistency and verifiability metrics across three main approaches. The approaches investigate how performance may be affected through changes made to the contextual embeddings, transfer learning, or the introduction of memory networks for tracking entity states. We find that through the first two approaches, we saw a slight overall improvement in end story accuracy. However, the verifiability and consistency metrics have not improved. The third approach shows an overall decrease in all metrics, though this may be due to training limitations imposed by GPU memory limitations. We have provided insights into the results, and avenues for future work to further verify our results. Lastly, for the EntNet implementation, we have discussed where we think problems lie and how these could be further investigated given more time[1].

1 Introduction

While pretrained language models have achieved stellar results on a variety of natural language understanding tasks, it is still questionable the extent to which they truly *understand* language as humans do. In this work, we focus on the task of story plausibility classification known as TRIP (short for Tiered Reasoning for Intuitive Physics) and proposed by Storks et al. (2021). The task aims at measuring not only the ability of a model to classify a task as plausible of not, but also the ability of a model to *justify or explain* its prediction. More specifically, given two similar stories that differ only by **one sentence** and only one of which is implausible, the system is required to solve the following three tasks:

1. **Story Classification**: Identify the implausible story.
2. **Conflict Detection**: Highlight the two conflicting sentence in the implausible story.
3. **Physical State Classification**: Identify the underlying physical states causing the conflict.

In a general sense, The TRIP benchmark aims to measure an NLP system's coherence in high-level plausibility prediction by measuring if the prediction can be supported or verified by low-level evidence.

In this project, our goal is to improve the performance on the TRIP task. As shown in (Storks et al., 2021), state-of-the-art language models do not seem to perform well on the task especially in terms of both **Verifiability** and **Consistency**. Therefore, in this project, we aim at improving the performance in terms of such metrics. We experiment with **three** different approaches: Commonsense Contextual Embeddings, transfer learning with relevant reasoning tasks, and improving Entity classification with memory networks.

Our report is organized as follows: Section 3 describes all our proposed techniques, Section 4 presents our results and discussion, Section 5 is our conclusion, and Section 6 shows the contribution of each team member.

2 Previous Work

The existing baseline for this problem proposed in the TRIP paper (Storks et al., 2021) uses a series of incremental neural networks to learn the associated tasks from the dataset. Given an input pair of stories, each sentence is then broken into entity-sentence pairs for each entity in the sentence. These pairs then pass through a pre-trained language model to produce contextual embeddings. The model learns the precondition and effect physical states for this entity-sentence pair from the en-

[1]Our code is available at https://github.com/mukhal/tiered-reasoning

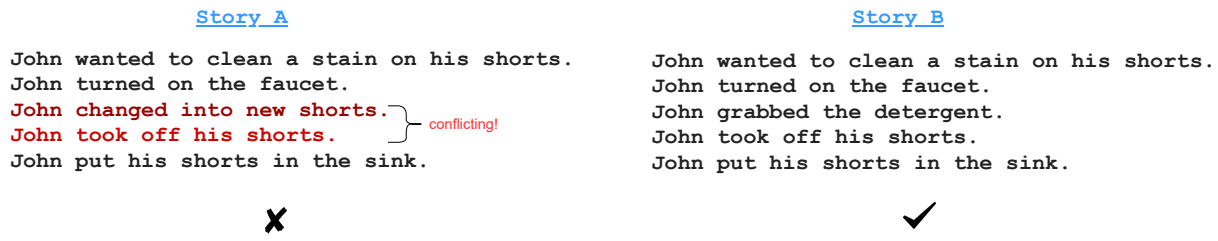

Figure 1: An example from the TRIP dataset (Storks et al., 2021). The story on the left is implausible due to conflict between the third and fourth sentences (highlighted in shades of red). The system is required to identify the implausible story, identify conflicting sentences, and identify underlying physical states causing the conflict.

codings and then utilizes all information to predict conflicting sentences and the overall story plausibility prediction.

A similar task, defined in the bAbI dataset (Weston et al., 2015), involves simple question and answer tasks that revolve around the knowledge of the entity states in a given series of sentences. While not directly related in terms of looking at the plausibility of sentences, it focuses on question answering about entities in a story. However, one would assume that knowledge of their states is required to answer questions about the entities, which is a related problem. Various tasks in the dataset test different types of reasoning relevant and related to TRIP. For example, temporal reasoning is tested in task 14 of bAbI. The dataset is machine-generated and only looks at the end-task performance, so it may be a concern that the models developed for the bAbI dataset do not truly understand the entity states to answer these questions. However, the implementations may still be of interest.

One such proposed model that succeeds in all 20 question and answer tasks on bAbI is EntNet (Henaff et al., 2017). In the EntNet model, inputs are initially encoded to a vector of fixed length. The authors suggest that any standard sequence encoder should work here. The inputs then pass through dynamic memory cells, which each contain vectors representing a key and value. The general idea is that each cell will learn to represent concepts or entities, and as such, the content is modified at the locations defined by the encoded input combined with the key vectors. When trained using 10k examples on the bAbI dataset, EntNet succeeds on all 20 tasks.

Other works have investigated the level of linguistical content captured by BERT. In one such paper, the authors investigate the linguistic structure learned by BERT and contained within the different layers of BERT(Jawahar et al., 2019). The lower layers of BERT perform better when predicting surface-level tasks such as word count and sentence length. In comparison, higher layers generally seem to perform better for tackling semantic tasks. The authors suggest that it may be the case that BERT can learn more complex information through further training but that this comes with a decreasing performance on surface-level tasks. The authors determine that the encodings produced by BERT capture surface, syntactic, and semantic linguistic signals. On the other hand, similar investigations suggested that these pre-trained language models encode syntax more than the higher-level semantic tasks(Tenney et al., 2019). However, the models still encode semantic representations, but the improvements over non-contextual representations are not as significant.

3 Approaches

In this project, we explore **three** orthogonal approaches to improve performance on the TRIP task. Our first approach is based on injecting commonsense reasoning ability into the pipeline proposed by (Storks et al., 2021) by means of a pre-trained commonsense language model. The second approach proposes that the classifier heads used for precondition and effect state classification are not capable of modelling the changes of entities over time and aims to implement a head designed for entity state tracking. We start by detailing our first approach (§ 3.1) and then we move to our second approach (§ 3.3).

3.1 Contextual Commonsense Embeddings

In this section, we first introduce COMET and then we describe how we integrate it into the pipeline proposed by (Storks et al., 2021). Pre-trained language models have been shown to lack commonsense knowledge required for various tasks (Klein

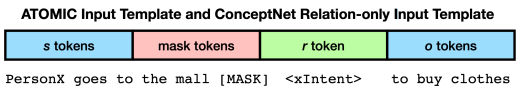

Figure 2: COMET pre-training procedure (Bosselut et al., 2019). As shown, COMET was pre-trained on ATOMIC to predict relation objects.

and Nabi, 2021). The main model proposed by Storks et al. (2021) relies on pre-trained BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) to compute contextual embeddings for sentences and entities, which are then used for both precondition and effect prediction. We hypothesize that we can obtain *better* contextual embeddings if we use models that are more fit for commonsense reasoning tasks. One such model is COMET (Bosselut et al., 2019). COMET is a transformer model pre-trained on a commonsense triplets to predict the object of a relation given both the relation and its subject. COMET was pre-trained on ATOMIC (Sap et al., 2019), which is a large commonsense knowledge base with different relations. Such relations include`xEffect`, which represents the effect of some action on some entity, and `xIntent`, which represents the intent of an entity when performing some action and so forth. Figure 2 shows the COMET pre-training procedure on ATOMIC. Figure 3 shows the full pipeline with COMET.

Our proposed approach is very simple; Instead of using a pre-trained language model to compute contextual embeddings of sentences and entities as done by Storks et al. (2021), we use COMET instead. While the approach seems simple, two important questions arise. First, unlike BERT and RoBERTa, we can not use the `[CLS]` token to extract embeddings since COMET (which is based on GPT-2) does not have special tokens. So how should we extract contextual embeddings from COMET? We can do max-pooling, average pooling, or introduce a new special token, all of which are valid options. The second question is how to encode both the sentence and the entities before feeding them to COMET.

To answer the first question, we experiment with *both* max-pooling and average pooling and show the results when using both methods. To answer the second question, we concatenate the sentence and the object separated by some special string. For example, given the sentence "Ann cracked an egg into the pan" with the entity "egg", the input to COMET becomes "Ann cracked an egg into the pan `<OBJ>` egg ", where <OBJ> is the *special string* used to designate the end of the sentence and the start of the entity.

We faced a few obstacles when integrating COMET into the TRIP pipeline. First, we needed to change data preprocessing to enable encoding and tokenization of the TRIP data using the COMET tokenizer. It is worth noting that COMET implementation[2] is based on an early implementation of GPT-2. So, we could not use *HuggingFace*. Instead, we had to write our own function that tokenizes and encodes the input sentence and entity as described above. Second, we had to match the rest of the pipeline with the output dimension of COMET, which required some code modifications.

3.2 Transfer Learning from Relevant Tasks

Since the size of `TRIP` is relatively small, one obvious direction is to train on more data that come from relevant reasoning tasks *i.e.,* transfer learning. One such task is Goal-Step Reasoning (GSR) (Zhang et al., 2020), which involves three sub-tasks: Goal inference, step inference, and step ordering. Goal inference is the task of selecting an appropriate goal from four possible choices given a specific action or step. Step inference involves selecting the correct step given the goal. Finally, step ordering is the task of predicting the correct order of two steps given a goal. Figure **??** shows an example from the WikiHow dataset.

We choose to pre-training on the goal inference task since intuitively, knowing the goal of specific action should be useful in identifying possible conflicts. For example, let us consider the sentence "John changed into new shorts" in Figure 1. Having some understanding of the goal of the first action, which could be to "go out", or "go to bed" is likely to *signal* a conflict when encountering the sentence "John took off his shorts". Therefore, we argue that the goal inference task is would be more helpful than other tasks. The GSR dataset we use is proposed in (Zhang et al., 2020) and comes from *WikiHow* and is relatively large ($\sim$ 190K examples). The GSR task could be used in either a pretraining or multi-task learning setting with `TRIP`. In this project and due to time constraints, we experiment *only* with pre-training. It is worth noting that, in a sense, what we are doing here can be thought of as *data augmentation*. Since the size of the TRIP

[2] https://github.com/atcbosselut/comet-commonsense

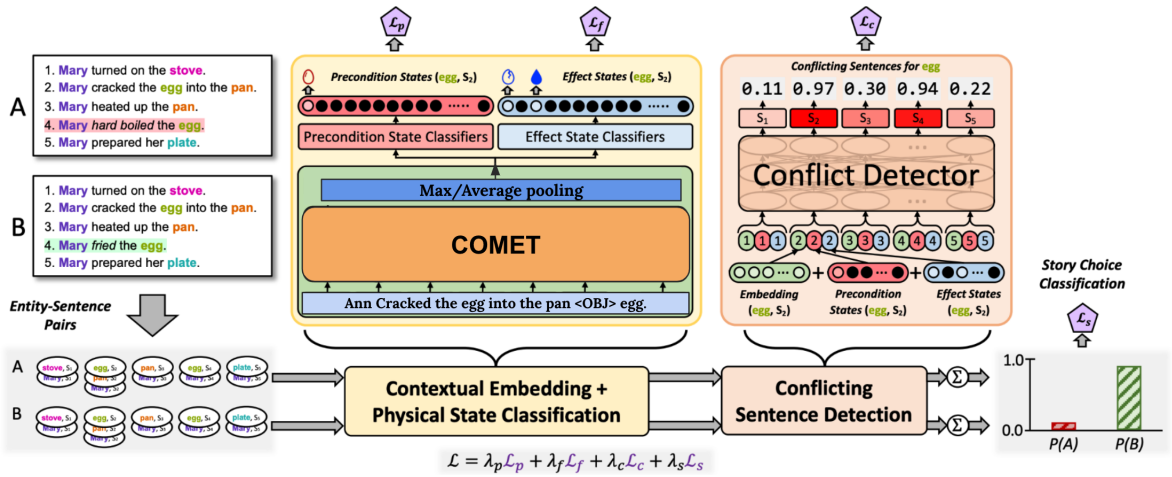

Figure 3: The pipeline proposed (Storks et al., 2021) with COMET used for contextual embeddings. The input sentence and entities are separated by a special string <OBJ> and the embeddings are extracted using either max- or average pooling.

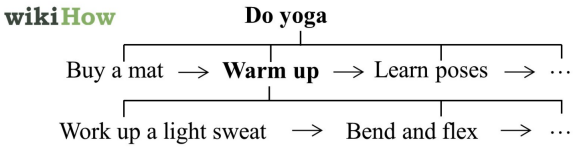

Figure 4: An example from the goal-step reasoning task proposed in (Zhang et al., 2020). The dataset consists of various steps and goals collected from WikiHow. Three tasks are proposed: predict step given goal (step inference), goal given step (goal inference), and correctly order steps. Here, we focus on the goal inference task and use it to pre-train our contextual embedding model before fine-tune it on TRIP.

dataset is relatively small, it makes sense to use more data from relevant sources, whenever possible.

3.3 EntNet

Given that work has shown that syntactic and semantic linguistic signals are captured by BERT (Jawahar et al., 2019), we hypothesize that there is a strong enough representation to capture the precondition and effect states of the entities. However, even with the loss associated with story choice omitted in the TRIP paper, the verifiability remains at only 10.6% with RoBERTa (Storks et al., 2021). As such, it may be that the classifier used on the embeddings cannot fully utilize the linguistic content in the embeddings. A starting point to investigate this may be to implement a classifier based on the Recurrent Entity Network (EntNet) (Henaff et al., 2017). We had considered that alternatively, providing the full story context to BERT may improve results by providing a richer contextual embedding if the EntNet approach fails. However, the EntNet approach was challenging to implement and required significant changes to the data pipeline, and as such we only pursued the EntNet implementation.

3.4 TRIP Data Preprocessing Changes

EntNet is not designed for processing data in the format that TRIP provides data. Since EntNet expects a whole story to be input and then a query string to answer questions about entities in the story, we need to make modifications such that TRIP data can be input in this manner. Since we want to recall information about the attributes of particular entities, we decided to use an encoding of the entity name as the query string to provide to EntNet. The idea is that the EntNet classifier heads will learn to associate different blocks of memory with different entities and retrieve the relevant attribute knowledge using the query string of the entity name.

We considered modifying the contextualized embeddings only to contain sentences since the entity query string could replace the association of entity-sentence pairs. However, due to memory limitations, we had to use the same embedding for the full network, and the EntNet heads. We didn't want to make significant changes to the base model pipeline, so we used the original entity-sentence pair embedding for the full model, and the EntNet heads. Initially, we transposed the input such that the num_sents dimension is the first dimension since we need to pass the data in sentence-by-sentence to the EntNet heads. However, initial train-

ing results suggested this may have changed the inputs far too significantly for the rest of the model. Intuitively we thought this might improve sentence-by-sentence input as it may preserve the ordering of the sentence. The overall model performance seemed to perform poorly, and we thought it might be due to the reshaped inputs, so we also tested the model using the original embedding shapes, passing a `view` of the embeddings to the EntNet heads.

3.5 EntNet Changes

Since the baseline TRIP model uses a classifier head for each attribute for both precondition and effect classification, we attempted to implement a head based on the EntNet model to replace these. The EntNet model will read a story and then answer questions on the story using an input query string. The model retains knowledge of the world state through the use of gated memory cells(Henaff et al., 2017). Since the cells update after each sentence, we speculate that we need to make predictions after each sentence passes through the EntNet model. The world state knowledge retained will change as each sentence passes through since EntNet is designed to answer questions at the end of the story. We thus use the modified embeddings outlined in Section 3.4 by feeding a sentence through the memory cell and then taking the hidden states and making a prediction through the EntNet output module after each sentence to make predictions about entity states at that sentence.

Ideally, we would have made the number of memory cells in the EntNet heads the same as the number of entities in the story and used a hidden state of the same size as the encoded inputs. However, this proved infeasible as the models' memory usage became far too large. As such, we added a linear layer to project the size of the encoded input and query string down to a hidden size of 32 and used 4 memory cells in each EntNet head.

3.6 CUDA Memory Limitations

The EntNet model is heavy and better suited as a full model rather than as multiple heads on another model. Having a total of 40 EntNet heads in as the classifier heads resulted in significantly higher memory usage. In order to reduce memory usage such that testing the model was possible, we had to introduce several limitations to the training pipeline.

As discussed previously, we had hoped to use a memory cell per entity and a hidden size the same as the dimension of the word embeddings. However, this was very infeasible. The embeddings for the sentences had to be shared with the input to the rest of the model, while it likely would have been better to use separate smaller embeddings for the EntNet heads. Performance may have been better if the original embeddings were not changed for the base model, and instead, a simpler input encoding, such as Bag-of-Words, was added to the dataset and used for EntNet heads.

The memory usage was still too high after reducing the model size to a hidden dimension of 32 with 4 memory cells per head. To further investigate, we implemented code to monitor the GPU memory usage and changes in the Python garbage collector throughout the training code to identify where PyTorch tensors were being created and causing the memory to run out. Through this, we determined that the `AdamW` optimizer was doubling the memory usage for the model since the added EntNet heads increased the number of model parameters significantly. The optimizer had to be changed to reduce memory usage, and by changing it to Stochastic Gradient Descent, the memory usage was finally low enough to train.

The code could run following the size reduction, shared input embeddings, and changing the optimizer, but the training was extremely slow. We implemented FP16 Mixed Precision training, which significantly increased the training speed though it is still a very long process. Training the maximum of 10 epochs specified in the base code takes over 6 hours, depending on the EntNet head parameters, so the EntNet implementation is trained for only 2 epochs. The limited-time of the project and the required time to train the model have reduced the amount of investigation performed into the model's performance.

3.7 Oversight in TRIP Baseline

While investigating the memory usage of EntNet, we noticed that the EntNet heads weren't being trained correctly, and the weights weren't changing. We verified that this is the case in the base TRIP model, and it is because the classifier heads are stored in a python list in the PyTorch model instead of a PyTorch `ModuleList`. This causes the parameters to be incorrectly tracked, and as such, the optimizer isn't aware of the parameters stored

in the python list. Changing the list doesn't significantly alter the TRIP paper results in the limited tests we had time to run, and we have notified the authors.

4 Evaluation and Discussion

4.1 Data

The dataset used will be the one collected in the paper (Storks et al., 2021). The data is a collection of stories, where each sample contains two similar stories, where one is implausible. There is some subjectivity associated with common sense and plausibility. The stories are written in a simple, declarative form and are grounded in the physical world to reduce subjectivity. Additionally, stories that were incoherent or contained unrealistic actions were removed from the dataset.

These stories are collected using Amazon Mechanical Turk. Then a single sentence is modified such that the sentence alone is plausible, but when combined with the context of the whole story, it makes the story implausible in the physical world. The resulting dataset contains a total of 675 plausible stories and 1472 implausible stories.

The stories are annotated with labels that cover a higher granularity than a simplified end task label. Aside from a label for the end task of determining which story is more plausible, annotations also exist to enable learning which pair of sentences conflicts to cause the implausibility. A final set of annotations describe the physical states of entities and the effect of each sentence on these states. An annotation of 20 physical attributes describes these states, and these attributes capture the contradictions between sentences in an individual story.

4.2 COMET Embeddings

Here, we show and discuss the results of using COMET to compute contextual embeddings as described in § 3.1. For simplicity, we show the results only on the test set of the TRIP benchmark. We run both training and evaluation using the code base provided by Storks et al. (2021) to ensure fair comparison with their published results. Since COMET is abased on GPT-2 small, it would be unfair to compare with RoBERTa-large and BERT-large since these contain many more parameters. Therefore, we also include the results from both RoBERTa-base and BERT-base. Table 1 shows the results when using different contextual embeddings models including COMET.

Based on the results, we have two observations. First, we can see that max pooling is performing *much better* that average pooling (∼16 accuracy points difference), which points to the significant effect that the way embeddings are extracted has over the task performance. Second, we can see that the best COMET model is outperforming both RoBERTa and BERT base in terms of accuracy but underperforms then in both verifiability and consistency. This could indicate that the way we are currently integrating COMET into the pipeline may not be the optimal approach to leverage the commonsense knowledge existing in COMET. A potential future work is to think of different ways to integrate COMET into the pipeline. One possible direction is similar to (Bosselut et al., 2021) who use COMET to generate a commonsense graph of the story entities in an *on-the-fly* fashion, and then reasoning is performed on top of the generated graph.

4.3 Transfer Learning from Goal-step Reasoning

Here, we explore the results of domain transfer from a relevant reasoning task. As explained in § 3.2, we choose the goal inference task (Zhang et al., 2020) as our pre-training task, which involves predicting the correct goal of a given action or step. Our approach here is fairly simple; We first fine-tune a language model (we experiment with both BERT and RoBERTa) on the goal inference task, then we fine-tune it on the TRIP data.

To fine-tune on the goal-inference task, we use the training set of the WikiHow data collected by Zhang et al. (2020). The training set includes ∼ 185K instances in the form of multiple-choice questions (MCQ), where given a step and four possible goal choices. The model is required to select the correct goal out of the four given choices.

To fine-tune on the WikiHow dataset, we use the same setting as in (Zhang et al., 2020). That is, we fine-tune for 3 epochs, with a learning rate of $5 \times 10^{-}5$ and a maximum sequence length of 200 tokens. Table 2 shows the performance with and without pre-training on the WikiHow dataset for both RoBERTa and BERT base models. We can see that the RoBERTa (WikiHow) model is performing comparably to vanilla RoBERTa while *outperforming* it in accuracy. On the other hand, we see that the BERT (WikiHow) model is performing much worse than the vanilla BERT. One

Model	Accuracy (%)	Verifiability (%)	Consistency (%)
Random	49.5	0.0	10.7
BERT-Large	70.9	8.3	21.9
RoBERTa-Large	75.2	5.7	18.8
RoBERTa-Base	72.4	6.3	22.50
BERT-Base	72.1	4.8	16.52
COMET (average pooling)	58.9	0.0	1.2
COMET (max pooling)	74.35	2.84	11.7

Table 1: Results on the TRIP test set with different contextual embedding models including COMET.

possible explanation for this drastic drop in performance with BERT is that BERT may be more susceptible to catastrophic forgetting than RoBERTa, and therefore fine-tuning on WikiHow could have erased some of the pre-training knowledge that BERT originally had, causing it to perform much worse on other tasks. Further analysis, however, is needed to confirm this.

In the end, we see that pre-training on the goal-step reasoning task did not provide much benefit as we initially expected. We argue that this may be attributed to the nature of the GSR task, which is framed as in an MCQ fashion. The task may be *too easy* for the model that performing well on this task does not really imbue the model with more reasoning capabilities. We hypothesize that leveraging the WikiHow data in creating a better, more informative, pre-training task.

4.4 EntNet Discussion

The EntNet results (Table 3) show significantly worse performance over the original TRIP baseline. We compare results between EntNet with reshaped and original embeddings, and additionally to the RoBERTa-large TRIP baseline model. All results are from the omit-story-loss method of training. Initially, we thought this might be due to reshaping the input embeddings such that the data was no longer coherent enough for the model to learn. However, the poor performance has carried over even when using the original embeddings; in fact, consistency is worse with the original embeddings (0.05% and 0.00%). I would suspect this is due to using the Stochastic Gradient Descent optimizer over the AdamW optimizer in the original code, as the base model shouldn't be affected by other implemented changes. Although only over two epochs, the training output shows a slight improvement, so modifying the learning rate or training for longer may be all that is required.

The EntNet heads may not be performing well for several reasons. As discussed above, modifications to the model outlined in the paper to work with the TRIP task may have reduced performance. Additionally, the small hidden size and the small number of memory cells may be causing problems. Ideally, EntNet would run with as many memory cells as entities in the stories. The model is also only trained for two epochs, and we may see some slight improvements with more training or a better optimizer.

While this approach set out to investigate whether verifiability could be improved by implementing memory into the state classifier heads, further experiments are needed to make a full conclusion. Given more time, there are a few avenues that we would take to conclude. We think that instead of using small EntNet heads as classifiers, it would be worth implementing a larger EntNet model and using this to predict for all attributes instead of one head per attribute. We also think it would be worth investigating using a separate embedding solely for the EntNet section, such as the encoding outlined in the original EntNet paper. This implementation may also use less memory than having several smaller EntNet heads.

Additionally, if the memory usage can be resolved, it would be valuable to rerun the experiments with the AdamW optimizer and the maximum ten epochs, as defined in the original Jupyter Notebook TRIP configuration. It would also be worth looking at other loss configurations as we only tested the *omit story choice loss* configuration.

An even more straightforward approach would be to investigate varying the learning rate of the SGD optimizer or training for more epochs given more time.

Model	Accuracy (%)	Verifiability (%)	Consistency (%)
Random	49.5	0.0	10.7
RoBERTa-Base	72.4	6.3	22.50
BERT-Base	72.1	4.8	16.52
RoBERTa (WikiHow)	74.9	5.1	21.4
BERT (WikiHow)	38.2	0.0	0.0

Table 2: Results on the TRIP test set with different contextual embedding models including COMET.

Model	Accuracy (%)	Verifiability (%)	Consistency (%)
RoBERTa-Large	73.6	10.6	22.4
EntNet (Reshaped Embeddings)	40.7	0.0	0.05
EntNet (Original Embedding Shapes)	40.2	0.0	0.0

Table 3: Results on the TRIP test set comparing EntNet with different embedding shapes and with the RoBERTa large. All results from the *omit story choice loss* configuration.

5 Conclusion

In this project, we explored three approaches to improve verifiability and consistency for TRIP. The first approach relied on commonsense embeddings extracted from a pre-trained commonsense language model. The second approach relied on transfer learning by pre-training on a relevant task, namely Goal-step reasoning. The final approach relied on Recurrent Entity Networks (EntNet) for precondition and effect classification of entities. The first two approach show performance gains in terms of accuracy but performance drop with respect to both verifiability and consistency. While the EntNet approach has led to significant decreases in performance, we believe this is due to limitations that had to be implemented to the training pipeline to conserve GPU memory. We have outlined future avenues to confirm this and described how the model and training could be modified to produce more reliable results given more time.

6 Work Division

Table 4 shows the contribution of each team member to the project.

Member	Contribution
Muhammad Khalifa	Implementation and evaluation of: COMET Embeddings, domain transfer from WikiHow - Writing of sections (§ 1, § 3.1, § 3.2, § 4.2, § 4.3)
James Tavernor	Implementation and evaluation of EntNet model - Writing of sections (Abstract, § 2, § 3.4, § 3.5, § 3.6, § 3.7, § 4.1, § 4.4.

Table 4: Work Division.

References

Antoine Bosselut, Ronan Le Bras, and Yejin Choi. 2021. Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. 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.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Tassilo Klein and Moin Nabi. 2021. Towards zero-shot commonsense reasoning with self-supervised refinement of language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8737–8743, Online and

Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

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

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with wikihow. *arXiv preprint arXiv:2009.07690*.