Common Sense Language Understanding - Improving Baseline for TRIP

Saurabh Budholiya
University of Michigan
srbh@umich.edu

Isaac Fung
University of Michigan
ifung@umich.edu

Abstract

With the advent of recent advances in Natural Language Understanding, large scale Language models(LM's) have started performing commonsense reasoning tasks similar to human beings. Tiered Reasoning for Intuitive Physics (TRIP) is an annotated commonsense reasoning dataset that enables multi-tiered examination of computers' reasoning process. In this project, we investigated some approaches to improve baseline performance of TRIP. Our Empirical results show that XLNet model performs similar to baseline results, while T5 model performs poorly. Ensemble models show their power in increasing accuracy of story classification, as they combine the best predictions from multiple models, but decrease the consistency as compared to baseline performance, which suggests a tradeoff between the two metrics.

1 Introduction

The knowledge of commonsense in everyday life such as 'Elephant can't be put in a fridge', 'Don't touch a hot gas stove' helps us navigate the circumstances in daily life smoothly, and creating systems which can mimic this commonsense reasoning similar to humans has been an active area for research for a long time (Gunning, 2018).

It is not that easy to create algorithms that can grasp natural language. Natural language has its own level of intricacy and ambiguity. Ambiguities can arise at various levels, including word meaning, syntactic structure, and semantic interpretation. Traditionally, Natural Language Understanding (NLU) systems have addressed ambiguities by utilising information from the textual context (for example, surrounding words and phrases), such as distributional approaches (Lenci, 2008)

For natural language modelling, Transformers have proven to be the most effective neural network architecture. Unlike recurrent neural networks (RNNs), which process text in a sequential manner, Transformers use self-attention to compute in parallel an attention weight for each word from the input text, allowing for significantly more parallelization than RNNs for large-scale model training (Vaswani et al., 2017)

Recent innovations in pretrained language models from (Devlin et al., 2018) and (Liu et al., 2019) have enabled NLP systems to perform closer to human beings in terms of language understanding. However, one significant distinction between human and machine text comprehension is that people have access to commonsense information while processing text in their minds, which allows them to make conclusions about things that are not explicitly stated in the text but are thought to be common knowledge (Modi et al., 2017).

(Storks et al., 2021) propose a new dataset called TRIP targeting physical commonsense reasoning. In TRIP, models are given two similar pair of stories, and they should attempt to identify which story is plausible according to commonsense, through detecting conflicts in either story and the physical state. In this project, we built up on this work by trying a few other approaches which we hypothesized might improve the baseline performance.

2 Related Work

Unsupervised learning has been used to discover simple commonsense relationships. For example, (Mikolov et al., 2013) show that by learning to predict adjacent words in a sentence, word vectors can be used to answer analogy questions such as: Man:King::Woman:?There have been various lines of research recently that have focused on commonsense reasoning. A lot of effort has gone into developing benchmark datasets that may be used to assess a system's performance on language processing and common sense reasoning tasks and spur the development of novel approaches to the problems. Several challenges have been proposed to evaluate machine intelligence based on commonsense, such

as The Winograd Schema Challenge (Levesque et al., 2012), which tries to resolve pronoun ambiguity using commonsense. (Bisk et al., 2020) evaluates the task of multiple choice text classification addressing physical commonsense. Another dataset, SWAG (Zellers et al., 2018), evaluates grounded commonsense inference by predicting the next scene given the current one. It is also a multiple-choice dataset with four other ways to continue the given description.

All of the above projects use multiple-choice datasets in which the user must select the correct option without providing any explanation as to why the machine chose that option. This raises doubts about the system's ability to comprehend the decision taken. None of the preceding works examines a direct grasp of commonsense by requiring a logical justification for a decision. (Wang et al., 2019) published a dataset that needs a system to select an unreasonable statement from a pair of statements and anticipate the correct rationale for its selection. They also used cutting-edge language models like BERT (Devlin et al., 2018). They have, however, experienced a drop in performance for explanation tasks. (Rajani et al., 2019) created a Commonsense Auto-Generated Explanation (CAGE) framework for commonsense QA challenge (Talmor et al., 2018). In its main technique of Reasoning, CAGE finetunes a language model based on the question and all possible answers. During training, the language model used Common Sense Explanations (CoS-E) as a referential explanation. The CoS-E dataset was created by hand and consists of reasons provided by users who were given the question, all answer options, and the proper label. This method was used to enhance a classifier in order to anticipate the answer to the multiple-choice question that was first posed.

There are several other works which evaluate common sense reasoning on other tasks such as on adversarial generation (Zellers et al., 2018), event validation (Wang et al., 2018), reading comprehension (Ostermann et al., 2018) etc. They assess whether a system has common sense by seeing if it can offer a valid answer when the input lacks such knowledge. The activities listed above do not evaluate commonsense validation directly, nor do they highlight the key aspect that must be considered in a commonsense language understanding procedure.

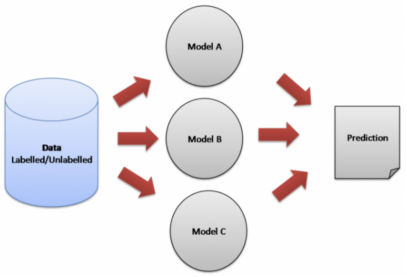

Figure 1: Maximum Confidence Ensemble Illustration

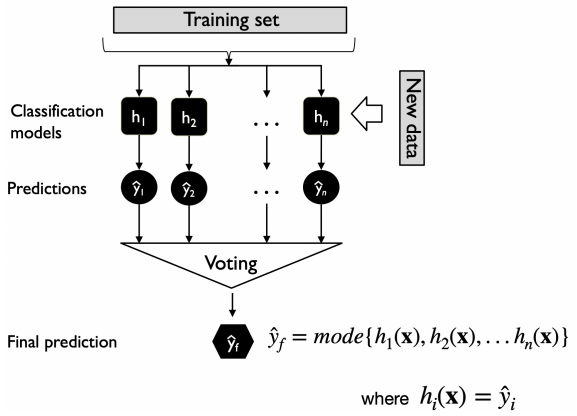

Figure 2: Maximum Voting Ensemble Illustration

3 Proposed Approaches

3.1 Adapting XLNet and T5 to TRIP

We built up upon the TRIP paper which used only BERT architecture models. We hypothesized that it might be interesting to see how pre-trained XLNet (Yang et al., 2019) and T5 (Raffel et al., 2019) Huggingface models perform on the TRIP dataset since both are also top performing models on the GLUE benchmark and they have differing architectures relative to the BERT-based models used in the paper. Since XLNet uses permutation language modelling, in which all tokens are predicted in random order, but BERT's masked language model only predicts the masked tokens, the model is able to learn bidirectional relationships and hence we believed that it would be able to handle word dependencies and relations better. T5 also uses a transfer learning paradigm, by encoding the different tasks as text instructions in the input stream, the T5 model addresses a wide variety of many-to-many and many-to-one NLP tasks in a unified manner, and hence we thought of experimenting with these two models for TRIP.

For XLNet, we needed to modify the TRIP codebase to support passing in XLNet's model parameters to train the model. Most of XLNet's model

parameters already existed in the TRIP codebase as the necessary code was already written to train other pre-trained Huggingface models. However, some parameters used different names which is why additional parameter names had to been read in order to successfully adapt XLNet.

While attempting to adapt T5 to TRIP, we discovered that we were required to pass in the decoder input ids for the base T5Model. Thus, we had to modify various functions called in the TRIP codebase to support generating and passing in the decoder input ids. We also tried training other types of T5 models like the T5EncoderModel and T5ForConditionalGeneration since they did not need the decoder input ids to be passed in, making it easier to get initial results.

3.2 Ensemble Learning for TRIP

After adapting XLNet and T5 to TRIP, we attempted to implement and evaluate an ensemble method's performance on TRIP. In (Liu, 2020), the author observed that ensemble learning performed well on the tasks since complementary models can correct each other to get better results. We were able to see which individual models perform the best against TRIP tasks based on the current baseline works (BERT, RoBERTa, DeBERTa) as well as the additional XLNet and T5 we adapted. We believed that it could be possibly effective and interesting to vary the ensembles used for different parts of the TRIP tasks similar to how Liu was able to achieve better performance on different SemEval subtasks. Thus, we trained and cached multiple models to evaluate the effectiveness of ensembling.

3.2.1 Maximum Voting Ensemble

For this approach, we first independently train each of the models that make up the ensemble. Using the the best performing version of each model from training, we call the functions needed to evaluate the models' performance on the testing dataset. This generates predicted labels for the various TRIP tasks such as story choice predictor and conflict detection. We iterate through the testing dataset predictions made for each task and compare the predicted labels of each model to generate a final prediction for the ensemble. For the story choice predictor task, since the predicted labels are binary, we set the ensemble's predicted label as the mode of the tasks that was the most chosen. This works well for when we use an odd number of models. In the case where we use an even number of models, we choose to make the ensemble's prediction label the same as a specified model. For both ensembles we used, we chose to specify this model as the RoBERTa baseline model since it performed the best overall as seen in the previous work by (Storks et al., 2021). For the conflict detection task, we proceeded along the similar lines, as the mode of the conflicts detected by the individual tasks, and in case of a tie, we labelled the conflict prediction of the ensemble model to be the prediction of RoBERTa baseline model, as explained above in the story choice classification task. The illustration for maximum voting ensemble is shown in figure 2 (Image Source - [1]).

3.2.2 Maximum Confidence Ensemble

For this approach, we modified the TRIP codebase to also return the probability of each of the predicted story task labels generated when a model is evaluated on the testing dataset. As a result, each model gave its prediction probability of class 0 and 1 respectively, by taking the maximum of it, we get the probability of the class predicted by the model. We iterate through the testing dataset predictions for each task and identify which model in the ensemble has the highest story probability (confidence). A simple visual illustration of this approach is shown in figure 1(Image Credit - [2]), in which the prediction is simply the model with highest class prediction probability. This model's predicted story label is then chosen as the ensemble's story label along with its value for 'valid explanation'. For the conflict task, the method for selecting the ensemble's prediction labels is the same as the previous ensemble where the conflict prediction of the ensemble model is the conflict prediction of the individual model with highest confidence.

4 Dataset Used

The only dataset we used for this project was TRIP dataset introduced by (Storks et al., 2021). Source code for running the evaluation pipeline and reproducing the results is available in [3].

5 Evaluation

We primarily experimented with the 'Omit Story Choice Loss' framework in (Storks et al., 2021) as

[1] https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/07$_{e}nsembles_{s}lides.pdf$

[2] https://towardsdatascience.com/ensemble-learning-using-scikit-learn-85c4531ff86a

[3] https://github.com/IsaacFung/eecs-595-final-project

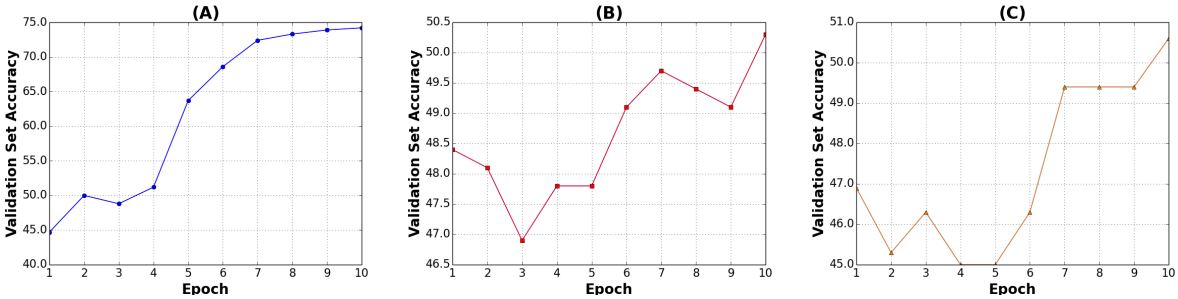

Figure 3: The blue line (A) displays the validation accuracy of the XLNet-base-cased model. The red line (B) displays the validation accuracy of the T5-small model. The orange line (C) displays the validation accuracy of the T5EncoderModel. All of these models shown above were trained with a learning rate of 1e-5 over ten epochs. They were found to be the best performing versions of the models after tuning the learning rate during training.

we are interested in classifying stories correctly and verifying them with coherent supporting evidence. This metric for evaluation is reported as 'Verifiability' by the authors. BERT and RoBERTa were the top two performing models as reported by the authors in TRIP dataset with respect to verifiability metric. On the other hand, accuracy is a traditional metric which simply measures the proportion of stories which are correctly classified. Consistency measures the proportion of examples in which stories are correctly classified and conflicts are correctly identified as well. Hence we show all the results of our new approaches in comparison to these two metrics.

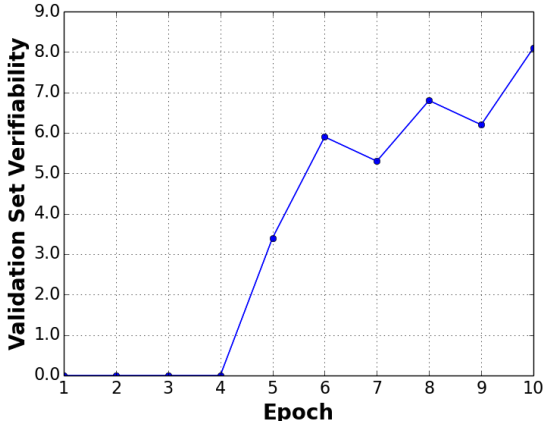

Figure 4: The validation verifiability for the XLNet-base-cased model trained with a learning rate of 1e-5 over 10 epochs (the model trained in Figure 3).

5.1 XLNet and T5 Results

For both XLNet and T5, we trained the models using a batch size of 1 in order to be consistent with the previous baseline models. We also trained all models over ten epochs.

5.1.1 XLNet

We trained XLNet with a learning rate of 1e-5 as that was the learning rate that the baseline RoBERTa model used. As we can see from tables 1 and 2, adapting XLNet for TRIP led to comparable results for the accuracy and verifiability with the existing baselines established by (Storks et al., 2021). Furthermore, in figure 3, it appears that XLNet's validation set accuracy begins to converges at epoch 8, reaching 74.2% accuracy at epoch 10. In figure 4, we can see that XLNet has a validation verifiability of 0% and jumps suddenly to 3.4% at epoch 5. It appears that the validation accuracy also jumps at epoch 5, demonstrating that the model likely needed to reach a certain accuracy level in order to start having verifiable results. Since the results were already comparable to the baseline, we did not focus on tuning the learning rate and instead focused on adapting T5 for TRIP.

5.1.2 T5

For both the T5EncoderModel and T5Model, we initially trained them with a learning rate of 1e-4 since the Huggingface documentation for T5 advised that for most problems using a learning rate of 1e-4 or 3e-4 was suitable for T5. We were originally using the T5-base model. However, while training the T5Model, we encountered a runtime CUDA error where we ran out of memory for the GPU we were using. Thus, we had to train a smaller instance of T5, T5-small. We ended up abandoning our approach of using the T5ForConditionalGeneration model and only used the T5EncoderModel and T5Model (T5-small) since we ran into difficulties with how TRIP was autogenerating labels. T5ForConditionalGeneration is supposed to convert the labels it is given to the

decoder input ids that can be passed in its T5 model. However, we ended up encountering incompatible shape errors when trying to train the model. Although we tried to modify the shape of the labels tensor to fix this, we realized it did not make conceptual sense and consequently abandoned using this version of T5.

For the T5EncoderModel, training it on a learning rate of 1e-4 resulted in a testing set accuracy, verifiability, and consistency of 43.3%, 0%, and 0% respectively. The T5Model also resulted in 0% verifiability and consistency, and had an accuracy of 38.2%. The highest validation accuracy for the T5EncoderModel and T5Model across all epochs were 49.9% and 49.7% respectively. Additionally, the validation accuracy and loss for both models kept on fluctuating between approximately 43% and 50% which suggested that the models were not learning anything over the ten epochs. Thus, we decided to try to re-train the model with some hyperparameter tuning. Since the accuracy and loss bounced up and down, this could possibly suggest that the learning rate is too large which is why we chose to adjust the learning rate to 1e-5 and train the models again.

After re-training the models, the accuracy and loss still bounced up and down over the ten epochs. However, there was also small increases in performance. The story prediction accuracy on the testing dataset for the T5EncoderModel and T5Model increased to 49.0% and 41.9% respectively as seen in table 1. The verifiability and the consistency of both models remained at 0% as displayed in tables 2 and 3. The validation accuracy over the ten epochs where these versions of the models were trained has been plotted as (B) and (C) under figure 3. While training over the ten epochs, the validation accuracy ranged from approximately 45% to 51%. The final epoch also resulted in the highest accuracy whereas the previous configuration's models with a learning rate of 1e-4 had their highest validation accuracy at epoch 3. As we can see, this configuration improved performance for accuracy but it still does not appear to be learning.

Evidently, even after re-training the model with a lower learning rate, the performance of these models is extremely poor. They performed worse than picking randomly between the two possible classes of 0 and 1, suggesting that both versions of the T5 model are not learning from the training dataset or they are learning incorrectly.

Model	**Accuracy(in %)**
XLNet	**74.6**
RoBERTa(**Baseline**)	73.6
BERT	73.9
T5Encoder	49.0
T5-small	41.9

Table 1: Accuracy comparison - XLNet and T5

Model	**Verifiability(in %)**
XLNet	7.4
RoBERTa(**Baseline**)	**10.6**
BERT	9.0
T5Encoder	0.0
T5-small	0.0

Table 2: Verifiability comparison - XLNet and T5

5.2 Ensemble Results

Due to time constraints, we only focused on the taking the ensemble of the story and conflict predictions and did not take into account the preconditions and effects. Since XLNet gave comparable results to the baselines established by (Storks et al., 2021), the RoBERTa model, we chose XLNet as one of the models. We didn't choose T5 as one of the models for our ensemble approach, because it gave poor results as displayed in Table 2. To add two other models to our ensemble, we also picked RoBERTa and BERT since they were strong baseline models. Together, these models made up the maximum voting and the maximum confidence ensembles.

5.2.1 Maximum Voting Ensemble Results

As indicated in Table 4, the maximum voting ensemble achieved a fairly higher accuracy of 77.2% for the story predictions compared to the baseline RoBERTa model's accuracy of 73.6%, indicating the effectiveness of this approach for increasing the accuracy. However, as seen in Table 5, the ensemble's verifiability was only 6.0% compared to the baseline model's verifiability of 10.6%. Additionally, the consistency metric of the maximum voting was also 6.0% which is significantly lower than the consistency of the baseline models. RoBERTa achieved a consistency of 22.8% and BERT achieved a consistency of 28.0%. Thus, it appears that there is a large tradeoff between increasing accuracy and decreasing verifiability and consistency when using this ensemble approach.

Model	Consistency(in %)
XLNet	23.4
RoBERTa(**Baseline**)	22.4
BERT	**28.0**
T5Encoder	0.0
T5-small	0.0

Table 3: Consistency comparison - XLNet and T5

Model	Accuracy(in %)
Max Voting Ensemble	**77.2**
RoBERTa(**Baseline**)	73.6

Table 4: Accuracy Comparison - Maximum Voting Ensemble

Model	Verifiability(in %)
Max Voting Ensemble	6.0
RoBERTa(**Baseline**)	**10.6**

Table 5: Verifiability Comparison - Maximum Voting Ensemble

Model	Accuracy(in %)
Max Confidence Ensemble	**76.3**
RoBERTa(**Baseline**)	73.6

Table 6: Accuracy Comparison - Maximum Confidence Ensemble

5.2.2 Maximum Confidence Ensemble Results

As indicated in Table 6, the maximum voting ensemble achieved a fairly higher accuracy of 76.3% for the story predictions compared to the baseline RoBERTa model's accuracy of 73.6%, indicating the effectiveness of this approach for increasing the accuracy. Also, as seen in Table 7, the ensemble's verifiability of 9.4% is relatively close to the baseline model's verifiability of 10.6%. Additionally, the consistency metric of the maximum voting was also 9.4% which is significantly lower than the consistency of the baseline models. RoBERTa achieved a consistency of 22.8% and BERT achieved a consistency of 28.0%. Thus, it appears that there is a large tradeoff between increasing accuracy and decreasing consistency when using this ensemble approach while the verifiability is relatively unaffected.

6 Discussion

6.1 Adapting XLNet and T5 to TRIP

6.1.1 Difficulties

Adapting XLNet and T5 to TRIP took longer than expected due to the heavier conceptual knowledge and engineering efforts required to integrate T5 to TRIP. Some of the model parameters for XLNet had different names compared to the parameters of models that TRIP was already integrated with. Thus, to integrate these parameters, we had to read these additional potential parameter names for XLNet. For T5, we also had to read in its specific model parameter names as well as do some modifications to various functions called during model training. The T5EncoderModel was able to run after logic to read in its specific model parameter names was added. However, the base T5 model, 'T5Model' required some additional decoder input ids that had to be passed for the specific model. Thus, we added logic to account for this throughout the training functions involved. While training the base T5 model, we encountered a runtime CUDA error where we ran out of memory for the GPU we were using. Thus, we had to train a smaller instance of T5, T5-small. We faced a lot of engineering challenges in getting TRIP to run with T5, spending a week and a half on it before proceeding to work on the ensemble methods. We had difficulties with learning the codebase in such a short amount of time which made it difficult to fully comprehend which areas of the codebase we should modify. We still attempted to modify multiple parts of the codebase to adapt T5 to TRIP and were eventually able to get it running, albeit with poor results.

6.1.2 Results Comparison

For XLNet, we were able to obtain a story prediction accuracy of 74.6% which is similar to the RoBERTa baseline model's 73.6% accuracy. We also achieved a verifiability of 7.4% and consistency of 23.4% which was also similar to the RoBERTa baseline model's scores of 10.6% and 22.4% respectively. However, we achieved significantly worse performance for both the T5EncoderModel and T5Model. We achieved story prediction accuracies of 49.0% and 41.9% respectively. The verifiability and consistency of both models was also 0%. Thus, it is evident that we had much greater success adapting XLNet to TRIP compared to T5. We suspect that both T5Models are not learning. This is because the validation accuracy of both models did not improve across epochs and instead bounced up and down in the 40% to 50% range as can be seen in Figure 3 (B)

Model	Verifiability(in %)
Max Confidence Ensemble	9.4
RoBERTa(**Baseline**)	**10.6**

Table 7: Verifiability Comparison - Maximum Confidence Ensemble

and (C). Both versions of T5 were initially trained with a learning rate of 1e-4,and since the accuracy and loss were bouncing up and down, it appeared that maybe that the learning rate was too large. Thus, we retrained both versions of T5 with a learning rate of 1e-5 and noted small improvements with the same pattern of bouncing up and down for the accuracy and loss.

6.1.3 Next Steps

It appears that there might be something fundamentally incorrect with our integration of the T5 models with TRIP that will definitely need further investigation beyond hyperparameters. Something that could be causing the issue is that our current integration is incompatible with how the labels for the previous work's models are generated in TRIP. They are currently generated for every model. Thus, we initially assumed that we could use the T5ForConditionalGeneration model since it is supposed to automatically generate the decoder input ids if given the labels. However, we ended up encountering incompatible shape errors. Although we attempted to manipulate the shape of the labels tensor to resolve this error, we realized it did not make conceptual sense and consequently abandoned using this version of T5. For the T5Model we did manage to get running, it is possible that the decoder input ids that we are generating and passing throughout training is still incorrect. As a result, more attempts should be made to modify this part of the codebase to address potential errors.

6.2 Ensemble Learning Approaches

6.2.1 Difficulties

For the ensemble learning paradigms, to simplify our analysis, we focused on applying ensemble techniques to only make predictions for story and conflicts, because for the remaining two attributes, 'preconditions' and 'effects', the predictions were in a form of nested dictionaries, which made it very difficult to apply any ensemble learning techniques given our timeframe.

6.2.2 Results Comparison

The maximum voting ensemble achieved the highest accuracy of 77.2%. However, both the maximum voting ensemble and maximum confidence ensembles outperformed the baselines in terms of accuracy in story classification as displayed in Table 4 and Table 6 respectively. The maximum confidence ensemble also achieved a verifiability of 9.4%, similar to the baseline of 10.6% as mentioned in Table 7 which somewhat validates our hypothesis that ensemble learning has the power to improve the performance in TRIP. The current performance is worse for the conflict prediction of the ensemble model, since we made an assumption that the model which has the highest confidence in classifying stories should be the model used to assign the conflict predictions to the ensemble model. We made this assumption for the consistency in an effort avoid overfitting the data, but it seems that this is might not be ideal.

6.2.3 Next Steps

The conflict predictions of the ensembles lagged behind in performance compared to the story predictions accuracy and verifiability. Therefore, we believe it could make sense to emphasize creating customized ensemble techniques that focus on the conflict predictions to see if it is possible to improve baseline performance in this task. Furthermore, it might also make sense to attempt another ensemble learning methods such as stacking that involves actually training the ensemble of the models together. We can approach stacking in a way like it is shown in figure 5 (Image Credit - [4]). The training set can be used in base-classifiers to train layers and produce a new feature matrix for the meta-classifier layer. The meta-classifier layer chooses a suitable classifier for final prediction classification. The maximum voting ensemble and maximum confidence ensemble approaches we tried both entail training all the models separately and combining their testing set predictions to generate the ensemble's predictions. Thus, it might be interesting to try using an ensemble approach in the training workflow itself.

7 Conclusion

Commonsense reasoning is a challenging task for machines, to understand nuances and complexi-

[4] https://www.researchgate.net/figure/The-architecture-of-the-stacking-ensemble-learning-In-the-base-classifiers-the-training$_fig1_335156833$

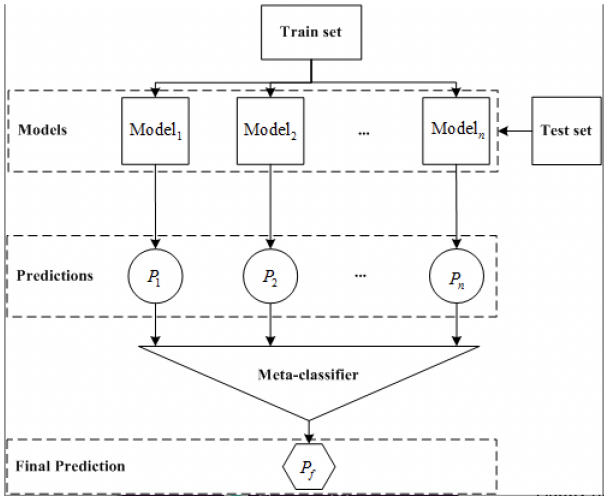

Figure 5: Stacking Ensemble Illustration

ties of events like humans and interpret their semantic representation. In this project, we studied the performance of XLNet and T5 models on TRIP dataset, in which XLNet gave us results similar to the baseline, while T5 performed poorly, which suggests us that XLNet might be employed in the future work of TRIP. Ensemble learning also achieves better performance in terms of accuracy and maintained similar verifiability at the expense of conflict detection performance.

8 Division of Work

This project required us to combine our efforts most of the times and we were committed to help each other. We faced a lot of engineering challenges, so we chose to brainstorm a lot of ideas together to get through them. Saurabh worked on training XLNet, with Isaac helping him though the troubleshooting of documentation of XLNet and BERT based models whereas Isaac worked on training the T5 model with Saurabh helping him on debugging the code, and understanding the intricacies of T5. For ensemble learning approaches, we both were involved in conceptualization of the idea that we undertook, and Saurabh worked on maximum voting ensemble paradigm and Isaac worked on maximum confidence ensemble paradigm.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

David Gunning. 2018. Machine common sense concept paper. *arXiv preprint arXiv:1810.07528*.

Alessandro Lenci. 2008. Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, 20(1):1–31.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Pai Liu. 2020. Qiaoning at semeval-2020 task 4: Commonsense validation and explanation system based on ensemble of language model. *arXiv preprint arXiv:2009.02645*.

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. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association for Computational Linguistics*, 5:31–44.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge. *arXiv preprint arXiv:1803.05223*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

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*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

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

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. *arXiv preprint arXiv:1906.00363*.

Su Wang, Greg Durrett, and Katrin Erk. 2018. Modeling semantic plausibility by injecting world knowledge. *arXiv preprint arXiv:1804.00619*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*.