Team 6 : EECS 595 Final Project Report

Ryan Deng
University of Michigan
zydeng@umich.edu

Yuqing Zhou
University of Michigan
zyq@umich.edu

Houming Chen
University of Michigan
houmingc@umich.edu

Abstract

In recent years, commonsense reasoning for language understanding has received increasing attention in the natural language processing field. In this project, we apply the BERT-based models to three benchmarks, CommonsenseQA, Conversational Entailment, and Everyday Actions in Text, to get knowledge of these benchmarks driving the progress in this area.

1 Introduction

Commonsense knowledge is knowledge unstated but is obvious to most humans beings. (Cambria et al., 2011). Although not stated, people can learn such knowledge through their daily experiences and from generalizing other commonsense knowledge (Speer et al., 2008). However, studying commonsense knowledge can be very difficult for machines. Although lots of research has been done, it is still a long way for machines to fully gain commonsense knowledge and apply them in Natural Language Inferences (NLI) (Storks et al., 2019).

In recent years, natural language processing (NLP) and commonsense reasoning have received increasing attention, and many benchmarks have been built to help on those studies (Storks et al., 2019). Those benchmarks are important to the NLI community because they provide good evaluations for various approaches and models and have encouraged more studies on NLI (Storks et al., 2019).

In this paper, we worked on three benchmarks related to commonsense reasoning: CommonsenseQA (Talmor et al., 2018), Conversation Entailment, and Everyday Action in Text (EAT). As recent studies on BERT (Devlin et al., 2019) and its variants have presented many state-of-the-art results in a variety of NLP tasks. We developed models mainly based on pre-trained BERT models and BERT variant models to do inference and prediction on these three tasks.

1.0.1 CommonsenseQA

CommonsenseQA is a common tool for evaluating how well a machine can understand human language and predict with it. People have used several different data sets in the NLP area, and there are four popular formats. For example, some datasets have the answer to be “yes” or “no”, or a unique answer choice for a multiple-choice insert to a paragraph. This opens a question: Can we train the commonsenseQA models to learn linguistic reasoning abilities? Our idea is simple: although the format of questions and the related information can differ across QA data sets, the underlying linguistic understanding and reasoning abilities are largely common. The underlying linguistic comprehension and thinking skills are universal to a large degree. So we start our project, by training models on a set of seed QA data sets of different formats, we construct a single pre-trained QA model, taking the natural text as input without using format-specific prefixes. Without using format-specific prefixes, we take natural text as an input. We use Bert and Roberta to predict the results for a given question and find BERT has the best accuracy.

1.0.2 Conversation Entailment

The text entailment task, determining if a hypothesis can be inferred from a given text, is an important component in NLI (Storks et al., 2020). But few works on the Conversation Entailment task have been done. Some research (Zhang and Chai, 2009, 2010) paying attention to automated entailment from conversational scripts used a probabilistic framework with an augmented representation of conversations which was a traditional way without deep learning. In our project, we use several modern models like BERT to predict whether a hypothesis is entailed by the given conversation.

1.0.3 Everyday Action in Text (EAT)

Some researches focus on hypothetical inferences of the language, which are defined as plausible inference by Davis and Marcus (2015). The Everyday Action in Text (EAT) is a plausible inference benchmark that provides short stories about some daily actions, and the task is to determine whether the story is plausible or not. If the story is implausible, the model should also determine the breaking point from which the story starts to be implausible.

In each sample, there is a short story that consists of 5 or 6 sentences. Without the story context, each sentence itself is consistent with the commonsense, but the entire story can be implausible when sentences are put together. For example, in the example shown below, the story is implausible since Tom cannot put on his gloves after he shredded his gloves.

```
{"story": [
"Tom put on his shoes.",
"Tom packed a suitcase.",
"Tom shredded his gloves.",
"Tom put his hat on.",
"Tom put on his gloves."],
"label": 0,
"breakpoint": 4,
"id": "train_3" }
```

Therefore, the task can be separated into two classification sub-tasks. The first sub-task is to predict whether the entire story is plausible. Label 0 is used to represent implausible stories and label 1 is used to represent plausible stories, so the task is to classify stories to $0/1$. The second sub-task is to predict the breakpoint of the story. -1 is used to represent plausible stories that have no breakpoints, and $1/2/3/4/5$ are used to represent the stories that start to be implausible from the $2nd/3rd/4th/5th/6th$ sentence. Then the task is to classify stories to $-1/1/2/3/4/5$.

In our project, we apply pre-trained BERT, RoBERTa, and DeBERTa models to do the two classification sub-tasks.

2 Computational Models

In this part, we are going to introduce what the data looks like, what models we use and what hyperparameters we choose for each task.

2.1 CommonsenseQA

We intend to apply BERT (Devlin et al., 2019) and also try RoBERTa (Liu et al., 2019) to see if we can get better accuracy scores.

Fetching data: We found that the "CommonsenseQA" dataset can be downloaded by calling $nlp.load\_dataset$ which is prepared and can be used instantly.

Create model: We create a BERT-style encoder transformer and the task head for the task. We use roberta-base as the model architecture in Transformers' AutoModels to further automate the model.

Processing Data: To input the data into the model, we first get the tokenizer corresponding to our model, then we convert raw text to tokenized text inputs.

Training Process: We set the constant seed to make sure every time We can use the same data to training the model. We also tried to change different hyperparameters to get the best model.

2.1.1 Data Preprocessing

The training dataset is downloaded from HuggingFace Datasets. It has 9741 samples and the test dataset has 1140 samples. There are four variables in the dataset that are *question*, *text*, *choices*, and *answerKey*. *AnswerKey* is used to verify. Here is an example:

Example
question: The sanctions against the school were a punishing blow, and they seemed to what the efforts the school had made to change?
text: ignore, enforce, authoritarian, yell at, avoid
choices: A, B, C, D, E
answerKey: A

2.1.2 Models

We use the pre-trained RoBERTa and BERT model from the Transformers package. Our target is to predict the best answer from the text, then return the corresponding choice.

Our best accuracy is 0.652 achieved by Bert when the learning rate is $1e-5$ and the batch size is 6.

2.2 Conversation Entailment

For the Conversation Entailment task, we try two types of models, BERT (Devlin et al., 2019) and its variant, RoBERTa (Liu et al., 2019).

2.2.1 Data Preprocessing

Before feeding the model with the data, we need to process the data. The training dataset has 520 samples. There are four different types of the hypotheses in the dataset: $belief$, $fact$, $desire$ and $intent$. The distribution of the dataset is shown as Table 2. In a sample, there are two items: one is to

Models	Learning Rate	Batch Size	Results
BERT	$1e-5$	2	0.60
BERT	$1e-5$	3	0.611
BERT	$1e-5$	4	0.639
BERT	$1e-5$	6	0.652
BERT	$1e-5$	10	0.641
RoBERTa	$1e-5$	2	0.601
RoBERTa	$1e-5$	3	0.615
RoBERTa	$1e-5$	4	0.632
RoBERTa	$1e-5$	6	0.649
RoBERTa	$1e-5$	10	0.631

Table 1: Results and hyperparameters for CommonsenseQA

indicate the speaker and another one is to show the utterance of the speaker, shown as follows.

Example 1 - Dialogue:
"speaker": "A",
"text": "I ripped the ligaments in my right ankle.",

"speaker": "B",
"text": "Gosh.",

"speaker": "A",
"text": "Yeah so,",

"speaker": "B",
"text": "Exercise is not supposed to do that to you."

One example of the format of the hypothesis is shown as follows.

Example 1 - Hypothesis:
"text": "SpeakerB thinks that exercise is not supposed to hurt anyone",
"tag": "h"

By observing the hypotheses in the dataset, we find that the subject such as $SpeakerB$ in *Example 1 - Hypothesis* does not appear in the utterances of $SpeakerA$ and $SpeakerB$. If we just concatenate the text of the utterances as the input, it may miss the information of the subjects which appears in the hypotheses, affecting the performance of the model. We add the information of the speakers at the beginning of each utterance. So the processed data looks like the following format:

SpeakerA says:"I ripped the ligaments in my right ankle."
SpeakerB says:"Gosh."
SpeakerA says:"Yeah so,"
SpeakerB says: "Exercise is not supposed to do that to you."

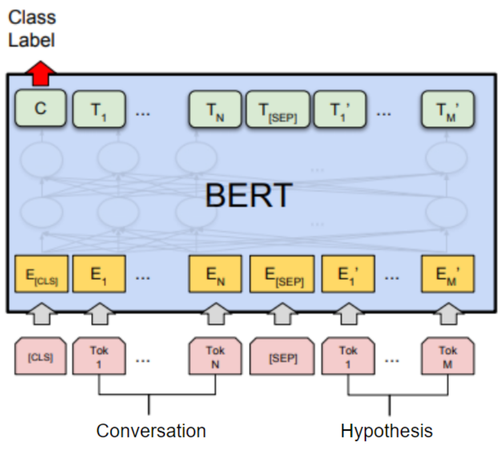

Figure 1: BERT for sequence classification task (Devlin et al., 2019).

Then we concatenate all utterances of the dialogue as a conversation that will be fed into the model together with the hypothesis.

Type of Hypotheses	**No. of Samples**	**Ratio**
belief	178	34.2%
facts	249	47.9%
desire	32	6.2%
intent	54	10.4%
unknown	7	1.3%
Sum	520	100%

Table 2: Distribution of the dataset for Conversation Entailment.

2.2.2 Models

We use the pre-trained model BERT (Devlin et al., 2019) and its variant RoBERTa (Liu et al., 2019). Our task is to predict whether the given hypothesis can be inferred from the conversation. This task can be classified as the sequence classification task and the corresponding model is shown as Fig. 1. BERT takes a concatenation of two segments that are sequences of tokens. In our task, the two segments are the conversation and the hypothesis. They are concatenated as a single input sequence to BERT with special tokens delimiting them (Devlin et al., 2019)(Liu et al., 2019), such as

$$[CLS], x_0, ..., x_{N-1}, [SEP], y_0, ..., y_{M-1}$$

$x_0, ..., x_{N-1}$ are the N tokens representing the conversation and $y_0, y_1, ..., y_{M-1}$ are the M tokens representing the hypothesis.

2.2.3 Implementations

We use a tokenizer to convert tokens to ids. Then we compute the maximum length of the sequences. We set a hyperparameter MAX_LEN that is larger than the maximum sequence length. If the sequence length is shorter than MAX_LEN, we pad the sequence. When using BERT (*'bert-base-uncased'*), we set the $MAX\_LEN = 512$. When using RoBERTa (*'roberta-base'* and *'roberta-large'*), we choose a value roughly larger than the maximum sequence length as MAX_LEN. Details are shown in Table 3.

Models	**MAX_LEN**
BERT	512
RoBERTa	440

Table 3: The maximum length set for each model in the Conversation Entailment task.

We have tested different splitting ratios and different batch sizes. Finally, we set the hyperparameter $split\_ratio = 0.2$, which means 20% of the dataset is used for validation and 80% is used for training. For the batch size, limited by Tesla T4 provided by Colab, the largest batch size for BERT is 8 and the largest batch size for RoBERTa is 4. We also use cross-validation to search for a better learning rate for BERT. These hyperparameters used for the final model are shown in Table 4.

Models	**BERT**	**RoBERTa**
Learning Rate	$5e-6$	$1e-6$
Batch Size	8	4

Table 4: Some hyperparameters for each model in the Conversation Entailment task.

2.3 Everyday Action in Text(EAT)

2.3.1 Split the Data set

The training data set contains 1044 samples. In each sample, there is a short story that consists of 5 or 6 sentences, a label indicating whether the story is plausible, and a break point indicating the first sentence after which the story stops making sense.

Among the 1044 samples in the data set, 522 of them are plausible and the other 522 of them are implausible. For each plausible story, there is an implausible story that is similar to it. Therefore, the 1044 samples can be grouped into 522 pairs. In each pair, the two stories are different by only one sentence. For example, the samples "$train\_3$" and "$train\_806$" are shown below. They are different by only one sentence but have different plausibility, so these two samples forms a pair.

```
{"story": [
"Tom put on his shoes.",
"Tom packed a suitcase.",
"Tom shredded his gloves.",
"Tom put his hat on.",
"Tom put on his gloves."],
"label": 0,
"breakpoint": 4,
"id": "train_3" }
```

```
{"story": [
"Tom put on his shoes.",
"Tom packed a suitcase.",
"Tom shredded his gloves.",
"Tom put his hat on.",
"Tom opened the front door and went out."],
"label": 1,
"breakpoint": -1,
"id": "train_806" }
```

Therefore, in order to make the evaluation better reflect the real performance of the model, we split the data by pairs. That is, after grouping the 1044 samples into 522 pairs, we split them into 422/100 as the train set and valid set. Therefore, the train set contains 844 samples and the valid set contains 200 samples.

2.3.2 Models and Implementation

Both the two tasks of EAT are classification tasks. While the first task is a binary classification: predicting plausible/implausible. The second task requires the model to classify the story into one of the six classes: $-1/1/2/3/4/5$ for the break point.

We applied BERT (Devlin et al., 2019) model and its variants RoBERTa (Liu et al., 2019) and DeDEBETa (He et al., 2020) to this task. We have trained 6 pre-trained models: bert-base-uncased, bert-large-uncase, roberta-base, roberta-large, deberta-base, and deberta-large. These models are all transformer models that contain transformer encoders. At first, the stories are padded with $MAX\_LEN = 480$ and then are tokenized into a sequence of tokens by the tokenizer of the pre-trained model. The transformer encoder then receives this sequence of tokens and encodes them to vectors. The first token '[CLS]' will be encoded as a class label that represents a sentence-level classification. Then we pass this class label vector into two single layers of a linear classifier. The first classifier gives a two-dimensional output for predicting the plausibility, and the second classifier gives a

6-dimensional output for predicting the breakpoint. Then the final loss will be calculated by adding up the cross-entropy losses for this two classification. Finally, We use the AdamW optimizer to optimize this loss function. The structure of the model is shown in Fig. 2.

After trying different hyperparameters, the hyperparameters that we chose are shown in Table 5. Other hyperparameters like dropout rates are consistent with the default configuration of the pre-trained model.

Each model is trained for 200 epochs, and after each epoch, the accuracy of sub-task 1 and the macro-F score of sub-task 2 are calculated on the validation set. The best performances on the validation set will be recorded to determine the performance of the model and to choose the best model to be our final model.

Model	batch size	learning rate
bert-base-uncased	16	3e-6
bert-large-uncased	4	3e-6
roberta-base	16	3e-6
roberta-large	4	2e-6
deberta-base	16	3e-6
deberta-large	2	3e-6

Table 5: Hyperparameters for EAT Model

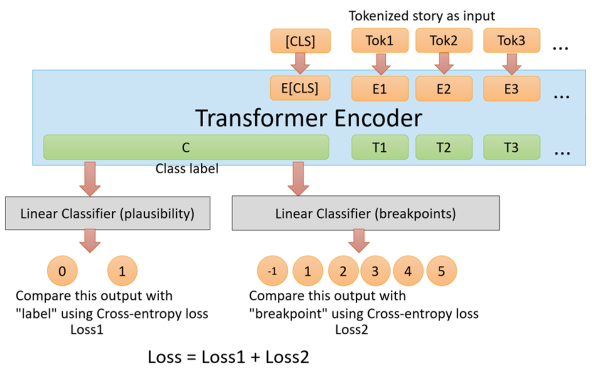
Tokenized story as input
[CLS]
Tok1
Tok2
Tok3
...
E[CLS]
E1
E2
E3
Transformer Encoder
C
T1
T2
T3
Class label
Linear Classifier (plausibility)
Linear Classifier (breakpoints)
0
1
-1
2
3
4
5
Compare this output with "label" using Cross-entropy loss Loss1
Compare this output with "breakpoint" using Cross-entropy loss Loss2
Loss = Loss1 + Loss2

Figure 2: Model used for the EAT task

3 Experimental Results

3.1 CommonsenseQA

Through trying different hyper-parameter, we found the best learning rate for BERT and RoBERTa is $1e-5$. After comparing 10 different models, we get the best result that is 65.2% achieved by BERT with a learning rate $1e-5$.

Models	BERT	RoBERT (large)
Learning Rate	$1e-5$	$1e-5$
Batch Size	6	6
Accuracy	65.2%	64.9%

Table 6: Results of CommonsenseQA

3.2 Conversation Entailment

By doing cross-validation, we find the best learning rate for BERT is $5e-6$. If we split the dataset into 5 subsets and use one of them in turn as the validation set, then the best average accuracy on the validation set over 5 experiments is 57.12% when training 7 epochs, and the best accuracy for a single experiment is 61% appearing at $epoch = 4$, as shown in Table 7.

We also try two kinds of RoBERTa for this task: *roberta-base* and *roberta-large* and find that *roberta-large* performs better than *roberta-base* and BERT. So we choose RoBERTa (*roberta-large*) as our final model for the Conversation Entailment task. As shown in Fig. 3, we fine-tuned RoBERTa on our benchmark for 60 epochs, but the best performance appeared at the $32th$ epoch, achieving an accuracy higher than 64%.

Models	BERT	RoBERTa (large)
Learning Rate	$5e-6$	$1e-6$
Batch Size	8	4
Accuracy	61%	64%
Epoch	4	32

Table 7: Best results and corresponding hyperparameters in the Conversation Entailment task.

Then, we further explore the performance of the fine-tuned RoBERTa model on different subsets of the validation set separated by the types of hypotheses. When the accuracy on the whole validation set is 60%, the accuracy on each subset is shown in Table 8.

3.3 Everyday Action in Text(EAT)

For the EAT task, the performance of the model is measured by the accuracy of sub-task 1 and the macro-F score of sub-task 2 on the validation set.

We trained 6 pre-trained models for the task: bert-base-uncased, bert-large-uncased, roberta-base, roberta-large, deberta-base, deberta-large. After fine-tuning the hyper-parameters, the best performance on the validation set are shown in Table 9.

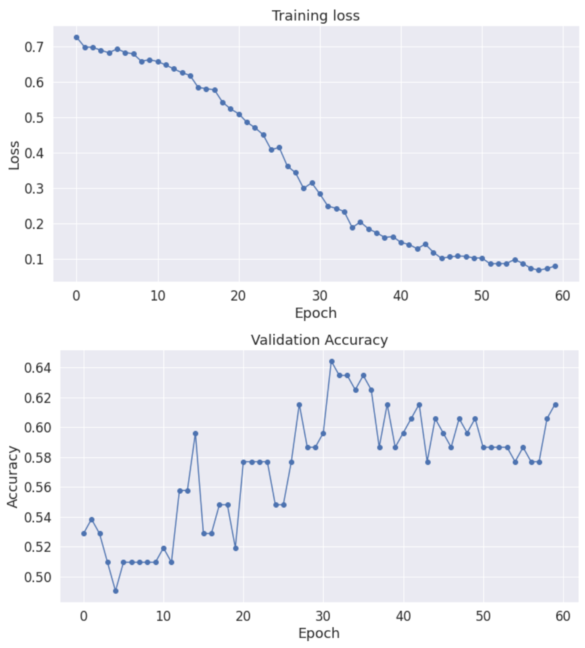

Figure 3: Training loss history and accuracy history on the validation set of RoBERTa (large)

Types	**Number of Training Samples**	**Number of Validation Samples**	**Accuracy**
belief	142	36	67%
fact	200	49	58%
desire	28	4	100%
intent	41	13	62%
n/a	5	2	0%
Total	416	104	60%

Table 8: Results of each type in the Conversation Entailment task using RoBERTa (large).

The deberta-large model gives the best result for both task 1 and task 2. Therefore, we decide to use the trained deberta-large model as our final model for prediction.

4 Discussion

In this section, we are drawing some conclusions from our results.

4.1 CommonsenseQA

The learning rate is an important hyper-parameter when training the model, it controls how much adjust the weights of the model. When we using a pre-trained model, we first use a large learning rate, and the result is not beautiful. Then we training to use a small learning rate, and the result is much better. Because high learning rates increase the risk of losing previous knowledge. Most of the pre-trained model has been well trained, use a small learning rate will not cause the weights too early and too much.

Batch size is a slider on the training process. Small batch size let the training process converges quickly at the cost of noise. On the other side, the large batch size lets the training process converges slowly with the accurate result of the error slope.

4.2 Conversation Entailment

As shown in Table 2 and 8, the dataset of Conversation Entailment is unevenly distributed. *Fact* is the most hypothesis among the dataset while *desire* has the least proportion except the unknown hypothesis type. However, the accuracy for the *desire* achieved 100% and the accuracy for the *fact* was the lowest among the four hypothesis types, achieving only 58%. The BERT-based model may be better at capturing the features of the data with *desire* and *belief* hypothesis types, leading to a good performance on those data, and it could be not good at capturing the characteristics of the data with *fact* hypothesis type. Another reason that accounts for the good performance on the subset with *desire* is a coincidence. It is very easy to achieve a higher accuracy or a lower accuracy by chance since it has only 4 samples of the *desire* type. The reason for the worst performance on the subset with the unknown hypothesis type is the same.

We also can find it is reasonable to set the split ratio relatively high, i.e., distributing more samples into the validation set, due to the small size of the whole dataset. If only having a few dozen samples for validation, it is very likely to achieve a high accuracy, which is unconvincing. Because this good performance is likely caused by chance and it may exist more samples whose hypothesis types are preferred by the model.

Another finding is the problem of overfitting. Due to the limited size of the dataset and the complexity of the model, it is easy to overfit. So we have to choose a small learning rate and a relatively large batch size to overcome the problem. For the BERT, the best performance usually occurs in 10 epochs. For the RoBERTa, as shown in Fig. 3, although there exists fluctuation in the validation accuracy's trend, we can see the best performance occurs at the $32th$ epoch. After that, the accuracy has a decreasing trend even though the training loss is decreasing, which is a mark of overfitting. So

Model	T1 Acc	T2 macro-precision	T2 macro-recall	T2 macro-F
Bert-base-uncased	0.7000	0.3674	0.3229	0.3359
Bert-large-uncased	0.7500	0.4793	0.3558	0.3800
Roberta-base	0.7100	0.3419	0.2963	0.3018
Roberta-large	0.7650	0.5276	0.3745	0.3972
Deberta-base	0.6850	0.4227	0.3014	0.3213
Deberta-large	0.8450	0.5494	0.5370	0.5338

Table 9: Performance of different models in the EAT task

we need to control the number of iterations during the training.

4.3 Everyday Action in Text(EAT)

4.3.1 Bias in the set and rare classes

Although the number of plausible samples and implausible samples are the same, in the implausible samples, the distribution of the breakpoints is not even. Among the 522 implausible samples, the number of stories having breakpoint at 1, 2, 3, 4 and 5 are 59, 87, 109, 262 and 5 respectively. The distribution is shown in Fig. 4

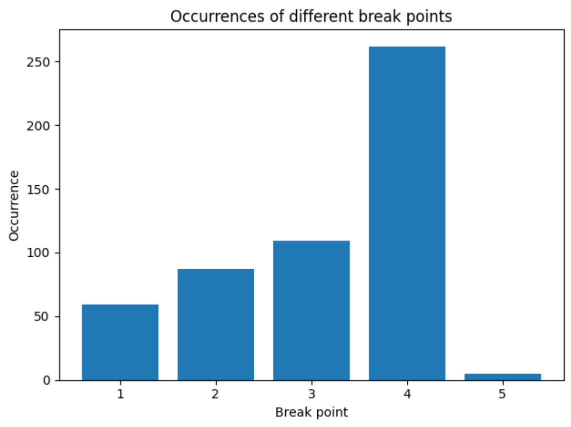

Figure 4: Occurrences of different break points of implausible stories in the EAT data set

We can see that the number of samples whose breakpoints are 4 is significantly greater than the numbers of samples having other breakpoints. Also, The samples whose breakpoints are 5 is extremely rare. Therefore, it is expected that the F-score for classifying 4 as the breakpoint would be greater than the F-score for classifying other breakpoints, because there are more samples for the model to whether a story has the breakpoint at 4. On the contrary, it is expected that the F-score for classifying 5 as the breakpoint would be very low, because there are not many samples for the model to learn when to predict that the breakpoint of the story is 5.

Table 10 shows the classification report of the DeBERTa-large model for sub-task 2. From the table, we can see that the F-score for classifying the breakpoint as 4 is the highest, which is higher than other classes, and the F-score for classifying the breakpoint as 5 is zero. This is consistent with our expectation and is caused by the bias of the data.

Since we are optimizing the macro-F scores of the sub-task 2, although the number of samples having breakpoints at 1, 2, 3 and 5 are less than the number of samples whose breakpoints are 4, the F-score of classifying the breakpoints as 1, 2, 3, or 5 are equally important as the F-score of classifying the breakpoint as 4. Therefore, it is very important to increase the F-score of classifying the breakpoint as 1, 2, 3, and 5.

One simple way to decrease the number of false positives of predicting breakpoint 5 is to check the lengths of the story. If a story has only 5 sentences, then the breakpoint can never be 5. Therefore, if the model predicts breakpoint 5 for a 5-sentences story, we can then change this predicted breakpoint 5 to breakpoint 4. However, since the model rarely predicts 5, this trick won't make much difference.

Another method is to change the loss function by giving greater punishment on misclassifying 1, 2, 3, and 5. However, this might cause the F-score of classifying the breakpoint as 4 to decrease, and then there would be a trade-off between the micro-F scores and the macro-F scores. We will try to change the loss function and see whether we could get better macro-F scores for sub-task 2 in future studies.

4.3.2 Inconsistency of the two classifiers

The task requires to maximize the sub-task 1 accuracy and the sub-task 2 F-scores at the same time. The two metrics are computed separately and added up together at the end. Therefore, in our model, we pass the class label produced by the transformer

breakpoint	precision	recall	f1-score	support
-1	0.8252	0.8500	0.8374	100
1	0.5625	0.7500	0.6429	12
2	0.6923	0.4091	0.5143	22
3	0.5625	0.4737	0.5143	19
4	0.6538	0.7391	0.6939	46
5	0.0000	0.0000	0.0000	1

Table 10: Classification report of the DeBERTa-large model for sub-task 2

encoded into 2 separate classifiers to get the best result. However, since the two classifiers work separately, it is possible that output can be inconsistent with each other, though they are consistent with each other most of the time. That is, the model might classify a story as plausible but also predicts a positive breakpoint at the same time; also, the model might classify a story as implausible but predict -1 as the breakpoint.

This won't a problem if we only want to get better numerical results on this benchmark. However, this will be a problem if the model is put in a real-life application because we won't expect the model contradict to itself.

This problem can be solved if we only use the breakpoint classifier, and predict that the story is plausible if the classifier predicts a "-1" breakpoint. However, this would make the accuracy for sub-task 1 lower as the model only optimize the cross-entropy loss of the breakpoint classification, but it is worth if the model will be used in an application.

4.3.3 Hyperparameters and Seed

When tuning parameters, we noticed that the pre-trained BERT based models are very sensitive to hyperparameters and even the seed. Many models can not be improved when the learning rate is large and sometimes, changing a seed might also cause the training loss to fail to decrease and the model to fail to learn. Therefore, it is important to try different hyperparameters and seeds when training these BERT based pre-trained models.

4.3.4 Error Analysis

Some researches classified commonsense into two types: intuitive physics, i.e. how the physical world works, and intuitive psychology, i.e. humans' motives and behaviors (Gunning, 2018).

From comparing the samples that our model succeed in predicting the right results and the samples that our model failed to predict the right result, we find that our model is relatively good at detecting the implausible story that is physically impossible, which only requires intuitive physics, like riding a disassembled bicycle or peel a banana that had already been peeled and sliced. However, our model performs relatively poorly when it requires both intuitive physics and intuitive psychology to determine the plausibility. Here, we show the following sample as an example:

{"story": ["John turned on the oven.",
"John put the cake in the oven.",
"John got the ice cream out.",
"John put some ice cream in a red bowl.",
"John put the red bowl in the oven."],
"label": 0,
"breakpoint": 4,
"id": "train_922" }

Our model failed to give the right prediction for this sample. In this story, it is physically possible for John to put ice cream in a bowl and then put the bowl in the oven. That is not normal, because no one wants to put ice cream in an oven.

In this example, the model not only needs to understand the physical properties of ice cream and oven, which are intuitive physics but also needs to understand that no one wants to eat a heated ice cream, which is intuitive psychology. Samples like this require the model to incorporate both intuitive physics and intuitive psychology to determine the plausibility, which is harder than applying only intuitive physics. This might be the reason that our model performs relatively poorly when it requires both intuitive physics and intuitive psychology to determine the plausibility.

This shows that although our models could understand some basic commonsense, there is still a long way for them to understand and apply more complicated commonsense in NLI.

References

Erik Cambria, Yangqiu Song, Haixun Wang, and Amir Hussain. 2011. Isanette: A common and common sense knowledge base for opinion mining. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 315–322. IEEE.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

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

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

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.

Robert Speer, Catherine Havasi, and Henry Lieberman. 2008. Analogyspace: Reducing the dimensionality of common sense knowledge. In *Aaai*, volume 8, pages 548–553.

Shane Storks, Qiaozi Gao, and Joyce Y Chai. 2019. Recent advances in natural language inference: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*.

Shane Storks, Qiaozi Gao, and Joyce Y. Chai. 2020. Recent advances in natural language inference: A survey of benchmarks, resources, and approaches.

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

Chen Zhang and Joyce Chai. 2009. What do we know about conversation participants: Experiments on conversation entailment. In *Proceedings of the SIGDIAL 2009 Conference*, pages 206–215, London, UK. Association for Computational Linguistics.

Chen Zhang and Joyce Chai. 2010. Towards conversation entailment: An empirical investigation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 756–766, Cambridge, MA. Association for Computational Linguistics.