Commonsense Validation and Explanation

Shilpu Srivastava
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
ssriv071@uottawa.ca

Abstract

In this paper, we present SemEval 2020 Task 4, Commonsense Validation and Explanation (ComVE), which includes work on three subtasks with the major objective of classifying a natural language statement as one that makes sense or does not make sense and also provide reasons for the latter. The first subtask requires the system to choose from two natural language statements containing similar phrases based on common sense. We have implemented subtask A using two approaches - using Google's Universal Sentence Encoder and a pretrained BERT model (Bi-LSTM coupled with BERT-based using hybrid pooling). The second subtask additionally asks the system to identify from three multiple-choice options, the appropriate reason for the sentence which does not make sense. We have used the pretrained BERT model to implement the solution for subtask B. The third subtask asks the participating systems to generate a reason for the sentence which does not make sense. For subtask C, we have fine-tuned a GPT-2 model on the custom ComVE dataset and have generated the reasons for the sentence that does not make sense. The models we built for these subtasks were able to achieve an accuracy of 88% for subtask A, 83% for subtask B, and a BLEU score of 5.27 for subtask C. According to the results, subtasks A and B trained using the pretrained BERT model exhibited good performance which explains how transfer learning can be used to leverage challenging NLU tasks like commonsense understanding and validation. However, for subtask C there still exists a huge gap between the text generated by humans and the system.

Keywords: ComVE, Google's USE, BERT, bert-based-uncased, LSTM, Bi-LSTM, hybrid pooling, BERT Tokenizer, GPT2, fine-tuning, GPU, CuDNNLSTM

I. Introduction

Over the years, the machine's ability to understand Natural Language has enhanced significantly. However, what remains challenging is its ability to identify sense in statements. The major idea behind successful text understanding and validation is the ability to read between the lines to get a concrete idea rather than a general one. What differentiates humans' interpretation of a text from that of a machine is the fact that humans can process the statement and judge whether the statement is reasonable and makes sense. For instance, the statements "He put a turkey into the fridge" and "He put an elephant into the fridge" can be easily differentiated by humans to identify the one which is plausible, but for machines identifying this difference is insignificant. Consequently, the idea is to explore options to increase the machine's ability in the commonsense understanding and validation arena. Introducing commonsense to natural language understanding is considered a novel direction of research to investigate the machine's ability in solving commonsense problems.

Various other existing tasks utilize the evaluation of commonsense understanding in other tasks such as adversarial generation [28], reading comprehension [29], coreference resolution [**?**], etc. Typically most of these tasks validate the understanding of commonsense indirectly by asking the model questions and expecting a correct answer when the input does not contain any related knowledge. These tasks do not directly evaluate the commonsense understanding but use the key concept of the same to accomplish their objectives.

To determine whether a sentence makes sense, additional context is required. In such scenarios, sentence embedding models come into the picture as they have the capability of including this additional information as they semantically encode the sentences. Various sentence embeddings that capture context are being used across numerous NLU tasks and commonsense validation is one among them. Various pretrained transformer models are fine-tuned for sentence embedding [27] and help to achieve state-of-the-art performance in problems that involve context. Google's Universal Sentence Embedding is also a sentence embedding model that transforms each sentence into a (1*512) dimensional embedding containing a lot of meaningful information about each sentence. Google's USE model is available in multiple variants - small, large, also multilingual and cant be used in tasks that require identification of the context of the sentences to accomplish a commonsense classification task. Using pre-trained models to embed the statements can also be used so that these embedded statements that are trained effectively with a large number of features can successfully capture the meaning of the sentences and therefore be used to detect against-commonsense statements.

Additionally, transfer learning is one of the most remarkable achievements in Natural Language Understanding which enables using the knowledge gained from one task (source task) to a distinct but similar target task. Transfer learning in a way is a workaround that saves us from reinventing the wheel and efficiently utilize the work that has been already implemented as part of other NLU tasks. Using the state-of-the-art pretrained neural network models like BERTs, ELMOs, OpenAI's GPT2 and GPT3 are some such examples of using transfer learning and can be used for commonsense understanding and validation problems.

As part of the ComVE challenge, the participants are asked to develop a system that can differentiate and explain counter-factual natural language statements. The ComVE challenge is divided into three subtasks. Subtask A is about commonsense validation where the objective is to differentiate statements that make sense from the ones that do not. For Subtask A we used two approaches - Google's Universal Sentence Encoders and a pretrained BERT model to perform the classification. For subtask B and C, the objectives are to explain the reason why a statement is against commonsense using distinct methods. Subtask B is about choosing the correct reasons from three options while Subtask C is around generating the reasons from scratch. For subtask B, we have used a pretrained BERT model and customized the model on the ComVE dataset. For subtask C, we have used the autoregressive GPT2 model and fine-tuned it on our custom dataset to generate reasons that explain why the contradictory sentence does not align with commonsense.

II. Literature Review

Natural Language Understanding frameworks have attracted more attention with the realization of the feasibility and significance of commonsense studies. As part of this project, we got an opportunity to review literature corresponding to commonsense reasoning. According to Davis and Morgenstern [2], automated commonsense reasoning has barely progressed and as opposed to what one may expect that researchers would be able to capture the reasoning aspects in various directions given the broad aspects of the problem, this has not been the case. Research seems to have been cluttered in few areas while other areas have remained dormant. Some areas of research that have been fairly stable over the last few decades include nonmonotonic reasoning, temporal reasoning, spatial reasoning [3], and theories corresponding to belief, desire, and intention [4].

However, with the advent of transfer learning by Yosinski et al. [5], as well as with the introduction of transformers by Vaswani et al. [6], various complex tasks in Natural Language Understanding arena gained significant progress and revolutionized the research and development process. One of the major milestones achieved was that transfer learning increased the ability of machines to understand and make sense out of raw statements. Furthermore, with the development of various sophisticated language models like BERT [7], ELMo [8] has also enhanced the machine's ability to interpret, make sense and validate the text.

As a result of commonsense studies being conducted, various datasets were prepared. The Choice of Plausible Alternatives (COPA) [9] focuses on events and results in which each question is given two choices to identify the appropriate cause/consequence of the premise. Many datasets were released as inspired by COPA, for example, the JHU Ordinal Commonsense Inference (JOCI) [10] dataset that gauges the possibilities of human responses after a certain situation on a scale of 1 (impossible) to 5 (very likely). A large-scale dataset called Situations with Adversarial Generations (SWAG) [11] used commonsense inference as the basis to determine what action or consequence happens to post a given situation. The main focus of the SWAG dataset is to assess pre-situations and post-situations to understand what caused the latter.

Various additional approaches derived from BERT like Robust Optimised BERT (RoBERTa) [12] and DistilBERT [13] used distillation to reduce the training and computation timings of the model while retaining its NLU capabilities. Various question answering datasets have also been released based on factual commonsense knowledge reasoning. A dataset called CommonsenseQA [14] provided a ConceptNet which has been used to create questions for the presented dataset. The dataset SQUABU [15] also depicted a simple test of commonsense and scientific questions. COSMOS QA [16] is also a large-scale dataset that consists of various multiple-choice questions based on contextual commonsense reasoning with 4 answers.

III. Task Description

As part of the Commonsense Validation and Explanation challenge (ComVE), the principal objective is to design and build a system that could differentiate natural language statements that make sense from those that do not make sense. This task is inspired and extended as part of the work carried out by Wang et. al (2019) [18] where the authors released a benchmark to measure commonsense with an explanation. The task is further divided into three subtasks. The first subtask A (Validation) is to classify two natural language statements with similar phrases to determine which one of the two makes sense and which one does not. The second subtask B (Explanation multi-choice) is to find the reason from three options that describe why the statement does not make sense. The third subtask C (Explanation generation) is to train the machine to generate the reasons which will be evaluated using BLEU scores as they will be assessed against the referential reasons.

Task A: Validation

Task: Which statement of the two is against common sense?

Statement1: I like to ride my chocolate.

Statement2: I like to ride my bike.

Task B: Explanation (Multi-Choice)

Task: Select the most corresponding reason why this statement is against common sense.

Statement: I like to ride my chocolate.

A: Chocolate is delicious and bikes are not.

B: Chocolate is a food, not a transportation unit.

C: My bike can't ride a chocolate.

Task C: Explanation (Generation)

Task C: Generate the reason why this statement is against common sense and we will use BLEU to evaluate it.

Statement: I like to ride my chocolate.

Referential Reasons:

1. Chocolate is a food, not a transportation unit.

2. You eat chocolate not ride it.
3. Can't ride chocolate.

IV. Dataset Description

The dataset used as part of this project was released as part of SemEval 2020 Task 4 - Commonsense Validation and Explanation [1] and we were able to access the data for all the tasks of this competition from the corresponding GitHub URL [17]. Each instance of the dataset comprises a set of 10 sentences distributed across the three subtasks: s1, s2, o1, o2, o3, r1, r2, r3. s1 and s2 are two similar statements that have the same syntactic structure and differ by a few words. One of these statements makes sense while the other does not. These two statements will be used for subtask A called validation where the participants are expected to identify the statement that does not make sense. For the statement that does not make sense among s1 and s2, three optional sentences o1, o2, and o3 that highlight the potential reasons why the statement does not make sense. These optional sentences will be used by subtask B named Explanation (multi-choice). Additionally, r1,r2, and r3 are three referential reasons which will be used to evaluate the BLEU score for subtask C named Explanation (Generation) where the participants are asked to generate the reasons for the non-sensible statement. The number of instances for the training, dev and testing dataset is shown in the Table I.

TABLE I
ComVE Dataset Instances

Dataset	No of instances
Training	10000
Validation	1000
Testing	1000

V. Technologies used

The original dataset consists of sentences and in order to perform modeling on the sentences, we needed to convert them into encodings or a vector representation so that the model can understand and learn the sentences better. We followed multiple approaches in order to achieve our objective for the three subtasks and used various technologies as part of the implementation of the same.

Some of the technologies that we incorporated as part of modeling the sentences for the three ComVE subtasks were - using ***Google's Universal Sentence Encoder*** to transform sentences into corresponding encodings, ***BERT tokenizer***, *adapt* ***bert-based-uncased*** *pretrained model on ComVE data*, ***GPT-2 finetuning***.

A. Google's Universal-Sentence-Encoder-Large

The Universal Sentence Encoder [20] is used to encode text into high 512 dimensional vectors and therefore carries meaningful information about the sentence that can be very useful for tasks like text classification, semantic similarity, and various other natural language tasks. This model is optimized in such a way that is works best for text which has a length greater-than-word, for example - phrases, sentences, and short paragraphs. Each sentence/phrase is converted into a meaningful 512-D vector by the USE model. The input can be variable length English text, but the output is always a consistent 512-D encoded vector. The USE is trained using two variants - one using the Transformer encoder and the other using the Deep Averaging Network (DAN) encoder [19]. The model already performs the best effort preprocessing of the sentences, so manual text preprocessing is not essential before providing the sentences to the model. For modeling the ComVE subtask A, we have used the Universal-Sentence-Encoder-Large model which is trained with a Transformer Encoder [20]. Fig. 1 shows the functioning of the USE model.

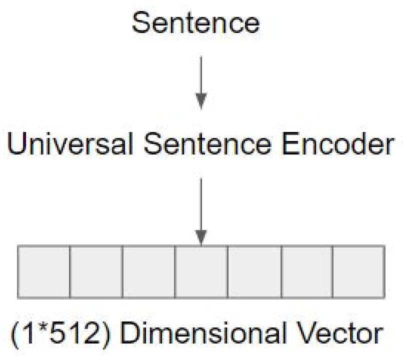
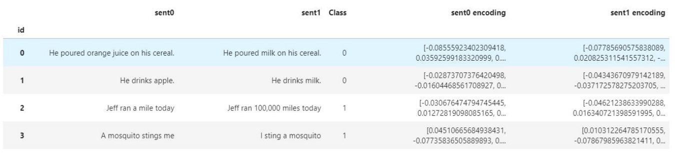


Fig. 1. Transformation of sentences into encoding using USE

Fig. 2 shows how ComVE sentences are converted to their corresponding encodings using the Universal Sentence Encoder Large model. Each instance of the attribute names 'sent0 encoding' and 'sent1 encoding' as shown in the figure is a 1-D array of dimension (1*512).



id	sent0	sent1	Class	sent0 encoding	sent1 encoding
0	He poured orange juice on his cereal.	He poured milk on his cereal.	0	[-0.08555923402309418, 0.03592599183320999, 0....	[-0.07785690575838089, 0.02082531541557312, -...
1	He drinks apple.	He drinks milk.	0	[-0.02873707376420498, -0.01604468561708927, 0...	[-0.04343670979142189, -0.037172578275203705, ...
2	Jeff ran a mile today	Jeff ran 100,000 miles today	1	[-0.030676474794745445, 0.01272819098085165, 0...	[-0.04621238633990288, 0.016340721398591995, 0...
3	A mosquito stings me	I sting a mosquito	1	[0.04510665684938431, -0.07735836505889893, 0....	[0.010312264785170555, -0.0786798596382141 1, 0...

Fig. 2. Transformation of ComVE sentences into encoding using USE

B. BERT Tokenizer

We have made use of the BERT Tokenizer to encode sentences in both subtasks A and B. To use a pre-trained BERT model, our sentences need to be converted into an appropriate format so that they can be fed into the pre-trained model to obtain the corresponding encodings. The Hugging Face's transformer [21] package provides various functions that enable this functionality. To preprocess the input data, a [CLS] token is added at the beginning and the [SEP] token is added to the end of each input text. We have made use of the built-in function of the BERT Tokenizer 'batch_encode_plus' that encodes both the sentences together for subtasks A and B and separates them by a [SEP] token. Additionally, to help the BERT model distinguish between the two sentences, the input is processed following the below steps:

- A [CLS] token is appended at the start of the first sentence and a [SEP] token is appended at the end of both the first and the second sentence.
- A segment embedding indicating whether the sentence is the first (Sentence A) or the second (Sentence B) is added to each token.
- A positional embedding showcasing the position of the token in the sequence is also added to each token.

Fig. 3 shows the BERT input representation of a sample ComVE sentence pair.

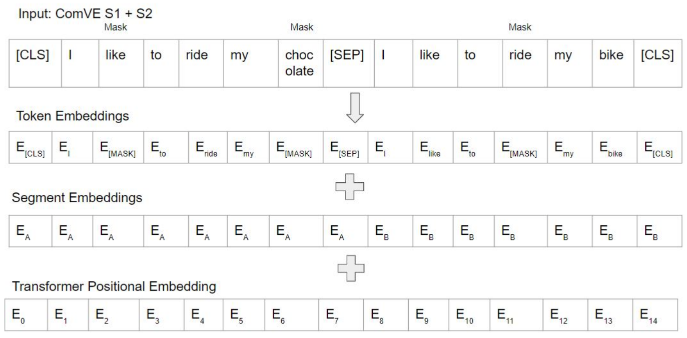


Fig. 3. BERT input representation for ComVE sentence pair. The input embeddings are the sum of the token embeddings, the segment embeddings and the position embeddings. Adapted from [7].

The BERT model requires a fixed length of sentence as input, so sentences that are shorter than the max length (128 in our case) need to be appended with empty tokens called paddings (represented by the token [PAD]) to make up the max-length. And finally, when the tokens are ready, they are converted to ids; in our implementation, the function 'batch_encode_plus' of the BERT Tokenizer returns the token ids and attention masks as encoded features for each of our sentence pairs. The sentences are padded and all the encodings are of the length 128. Fig. 4 shows how our sentence pairs from ComVE dataset are encoded together using the BERT Tokenizer.

C. *bert-based-uncased Pretrained model*

The BERT model is a transformer-based model that is pretrained on a large corpus of English data in a self-supervised manner, pretrained on raw texts, and no human labeling whatsoever [22]. It is an automatic process to generate inputs and labels from raw text. In order to train our model for subtasks A and B, we have used the pretrained 'bert-based-uncased' pretrained model trained in English. This is an uncased model, i.e. it does not differentiate between upper case and lower case words. It internally uses the masked language

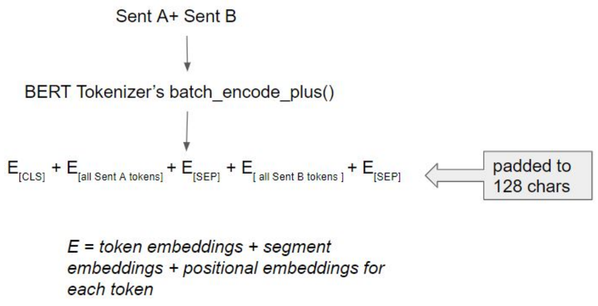

Sent A+ Sent B
BERT Tokenizer's batch_encode_plus()
$E_{[CLS]} + E_{[all\ Sent\ A\ tokens]} + E_{[SEP]} + E_{[all\ Sent\ B\ tokens]} + E_{[SEP]}$
padded to 128 chars
E = token embeddings + segment embeddings + positional embeddings for each token

Fig. 4. Transformation of ComVE sentences into encoding using BERT Tokenizer

modeling (MLM) technique where the model picks a sentence and randomly masks 15% of the words in the input text and runs it through the model with the objective of predicting the masked words [22]. An example of how the MLM technique works is shown in Fig. 5. This is different from other deep neural net models that usually model words sequentially or an autoregressive model that follows the idea of masking future tokens. With the MLM technique, the model is able to learn a bidirectional representation of the sentence. The BERT pretrained models are ideal to be fine-tuned for tasks that use (masked) sentences for decision making, sequence classification, token classification, etc. However, for tasks that involve text generation, using autoregressive models like GPT2/ GPT3 is ideal.

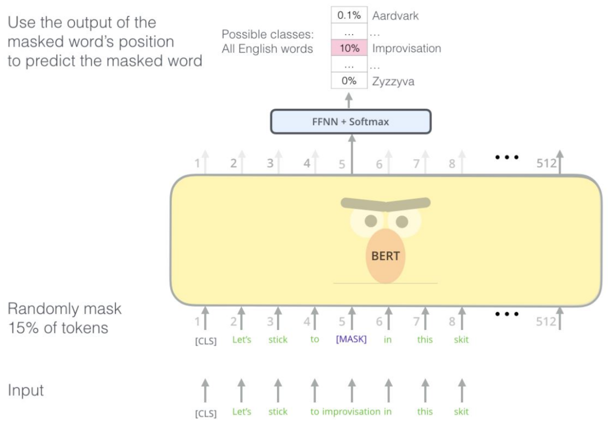

Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
10% Improvisation
0% Zyzzyva
FFNN + Softmax
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

Fig. 5. BERT's Masked Language Modeling (MLM) technique which masks 15% of words in the input and asks the model to predict the missing word. Retrieved from [24]

D. *GPT2 fine-tuning*

GPT2 is another popular transformer that is being widely used for Natural language text generation tasks. It is a large-scale transformer-based language model that is pre trained on a large corpus by scraping the web for around 8 million high-quality web pages [23]. Because of the huge dataset that it is trained on and the massive amount of features that it has (117M, 124M, 345M, 1.5B across the multiple variants of GPT-2), it exhibits competitive performance on various language tasks based on the pretrained knowledge without a requirement of training it explicitly. To enhance the performance of the GPT-2 model on the custom dataset, we decided to fine-tune it on the ComVE dataset and use it to generate reasons for subtask C. The high-level idea of GPT2 fine tuning for ComVE subtask C is shown in Fig. 6.

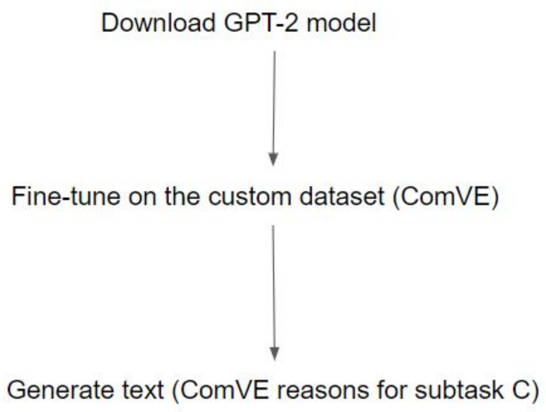

Download GPT-2 model
Fine-tune on the custom dataset (ComVE)
Generate text (ComVE reasons for subtask C)

Fig. 6. GPT2 fine-tuning high-level overview

VI. Design and Implementation

We tried to explore the technologies enlisted in Section V to implement the solutions to the three ComVE subtasks. Subtask A was implemented using USE and BERT; subtask B was implemented using BERT, while subtask C was implemented using the GPT2 fine tuning methodology.

A. Subtask A - Validation

The objective of subtask A of the ComVE challenge was to classify two natural language statements with similar phrases to determine which one of the two makes sense and which one does not.

Subtask A: Validation

Task: Which statement of the two is against common sense?

Statement1: I like to ride my chocolate.

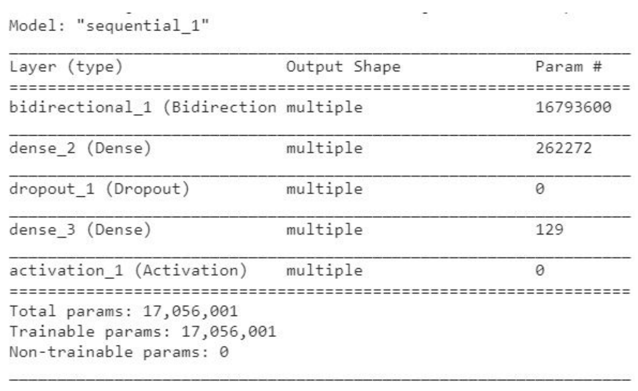
Statement2: I like to ride my bike.

We implemented subtask A using two methods: (1) using Google's Universal Sentence Encoder (2) using the pretrained BERT (bert-based-uncased) model. The below sections describe each of these approaches in detail:

1) ***Subtask A using Google's Universal Sentence Encoder****:* The first step to carry out the implementation of subtask A using Google's USE was to download the USE-large model from the tensorflow hub [20]. We downloaded the pretrained Universal Sentence Encoder available at the Tensorflow-hub in a REST API, and to optimize performance so that we do not have to call the module and session again and again, we called the module and created the session in the beginning and continued to reuse it for the rest of the statements. Using this method, we transformed our training, dev and test dataset into the corresponding USE encodings. As mentioned in Section V, the USE converts each sentence into a 512 D vector, hence each of our sentences were converted to a (1*512) dimensional array. We used the vertical stack (numpy.vstack) to stack up all the encoded Statement1 together and another vertical stack to stack up all the encoded Statement 2 together. Thereafter, we concatenated these two vstacks along the axis to have our final data ready where:

- X - shape (10000,1024) where 10000 is the number of instances in the training dataset and (2*512 = 1024) is the dimension of Statement1 and Statement2 encoded using Google's USE.
- y - shape (10000,) indicating the class label.

We passed these values to a Bidirectional LSTM model, and we also made use of GPU for faster training. Hence, we used the CuDNNLSTM library and trained our model. Fig 7 below shows the summary of our model and the layers used.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_1 (Bidirection multiple                  16793600
_________________________________________________________________
dense_2 (Dense)              multiple                  262272
_________________________________________________________________
dropout_1 (Dropout)          multiple                  0
_________________________________________________________________
dense_3 (Dense)              multiple                  129
_________________________________________________________________
activation_1 (Activation)    multiple                  0
=================================================================
Total params: 17,056,001
Trainable params: 17,056,001
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. Model Summary for Subtask A - USE model

We made use of the dev dataset for hyperparameter tuning of the model. Table II below shows the details around the hyperparameters we used to train our model.

TABLE II
Hyperparameters for Subtask A - USE model

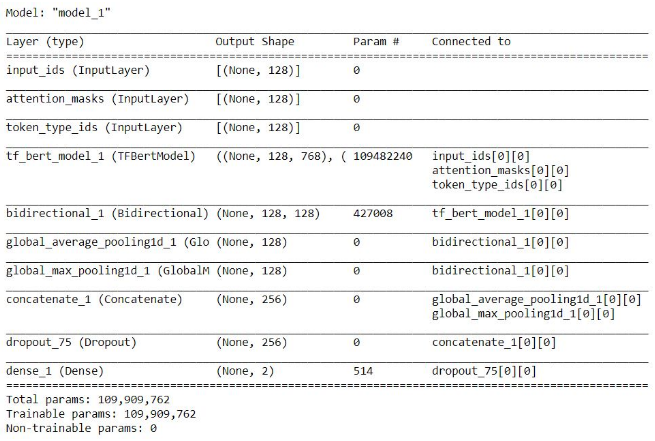
Hyperparameter	Value
No of epochs	350
Batch size	10000
Dropout	0.5
Activation Function	Sigmoid
loss	binary_crossentropy
optimizer	adam
metrics	accuracy

After tuning the parameters, we used this model to predict the class for the testing dataset and saved the results obtained into a CSV file which would later be passed to the scorer functions that have been made available in the ComVE SemEval Github repository [17].

*2) **Subtask A using pretrained BERT model (Bi-LSTM + BERT-base with hybrid pooling)**:* We have used the BERTSemanticDataGenerator() that helps us to generate batches of data in the ideal format that is required by the pretrained bert-based-uncased model. The BERTSemanticDataGenerator() function takes as arguments various parameters like the sentence_pairs, labels, batch_size, boolean to enable data to shuffle, boolean whether to include labels. The function returns tuples - ([input_ids, attention_mask, token_type_ids], labels) after passing the input parameters through the BERT Tokenizer's batch_encode_plus() function.

Specific to our ComVE dataset, we passed Statement1 and Statement2 (sentence_pairs), with their labels and a batch_size of 32 with a max_len of 128. Then we download the pretrained BERT model 'bert-based-uncased' and tokenize our statements using the batch_encode_plus() method of the pretrained BERT model. Then we convert the batch of encoded features to a NumPy array. Finally, we return a NumPy array containing the input_ids, attention_masks, token_type_ids], labels.

Thereafter, we load the pretrained bert-based-uncased model and freeze the BERT model to reuse the pretrained features without performing any modification. Then, we add the trainable layers on top of the frozen layer to adapt the pretrained features on our custom ComVE dataset. We generate a bi-lstm sequence output using the input_ids, attention_masks, token_type_ids and utilize a GPU for faster training (CuDNNLSTM). We further apply the hybrid pooling approach to the bi-lstm sequence output. We measure the training loss using the binary_crossentropy and use the accuracy metrics to evaluate our training and validation performance. We also enable multiprocessing with a total of 20 workers while we fit our model to enable faster training. We train the pretrained frozen model for 10 epochs. Post this step, we unfreeze the model and recompile it to add the optimizer and a small learning rate of 1e-5. After recompiling the model, we retrain the model for 10 more epochs after which we evaluate the model's performance on the test dataset. Fig 8 shows the model summary for the pretrained BERT model approach for subtask A.



```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_ids (InputLayer)          [(None, 128)]        0
__________________________________________________________________________________________________
attention_masks (InputLayer)    [(None, 128)]        0
__________________________________________________________________________________________________
token_type_ids (InputLayer)     [(None, 128)]        0
__________________________________________________________________________________________________
tf_bert_model_1 (TFBertModel)   ((None, 128, 768), ( 109482240   input_ids[0][0]
                                                                 attention_masks[0][0]
                                                                 token_type_ids[0][0]
__________________________________________________________________________________________________
bidirectional_1 (Bidirectional) (None, 128, 128)     427008      tf_bert_model_1[0][0]
__________________________________________________________________________________________________
global_average_pooling1d_1 (Glo (None, 128)          0           bidirectional_1[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_1 (GlobalM (None, 128)          0           bidirectional_1[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 256)          0           global_average_pooling1d_1[0][0]
                                                                 global_max_pooling1d_1[0][0]
__________________________________________________________________________________________________
dropout_75 (Dropout)            (None, 256)          0           concatenate_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 2)            514         dropout_75[0][0]
==================================================================================================
Total params: 109,909,762
Trainable params: 109,909,762
Non-trainable params: 0
```

Fig. 8. Model Summary for Subtask A - using pretrained BERT model

We used the BERTSemanticDataGenerator() to generate batches of the dev dataset which we utilized to tune the hyperparameters of the model. Table III below shows the details around the hyperparameters we used to fine-tune our pretrained BERT model for ComVE Subtask A.

TABLE III
HYPERPARAMETERS FOR SUBTASK A - USING PRETRAINED BERT MODEL

Hyperparameter	**Value**
No of epochs	10
Batch size	32
Dropout	0.3
Activation Function	Softmax
loss	binary_crossentropy
optimizer	adam
learning-rate	1e-5
metrics	accuracy
no_of_workers (multi_processing)	25

Based on the prediction probabilities obtained for class 0 and class 1, we assign the final classes to the test dataset and store the results to a CSV file which would be passed to the scorer function later to evaluate the accuracy of our prediction for ComVE subtask 1.

B. Subtask B - Explanation (Multi-Choice)

The objective of ComVE Subtask B (Explanation: multi-choice) is to find the reason from the three options that describe why the statement does not make sense. ***Task B: Explanation (Multi-Choice)***

Task: Select the most corresponding reason why this statement is against common sense.

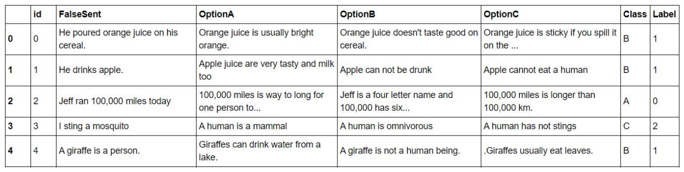
Statement: I like to ride my chocolate.

A: Chocolate is delicious and bikes are not.

B: Chocolate is a food, not a transportation unit.

C: My bike can't ride a chocolate.

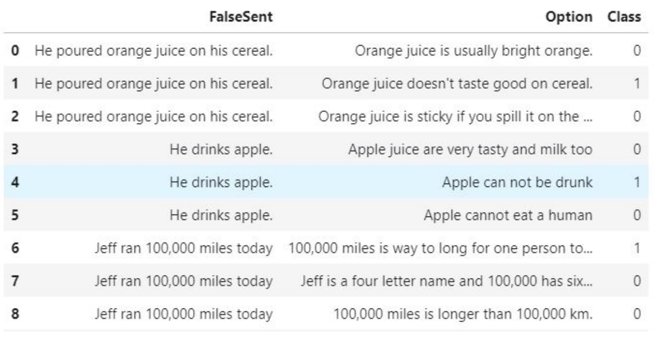
For this subtask, we transformed the format of the data so that we could provide input to the pretrained BERT model in pairs. Fig. 9 shows the original dataset for ComVE Subtask B.



	id	FalseSent	OptionA	OptionB	OptionC	Class	Label
0	0	He poured orange juice on his cereal.	Orange juice is usually bright orange.	Orange juice doesn't taste good on cereal.	Orange juice is sticky if you spill it on the ...	B	1
1	1	He drinks apple.	Apple juice are very tasty and milk too	Apple can not be drunk	Apple cannot eat a human	B	1
2	2	Jeff ran 100,000 miles today	100,000 miles is way to long for one person to...	Jeff is a four letter name and 100,000 has six...	100,000 miles is longer than 100,000 km.	A	0
3	3	I sting a mosquito	A human is a mammal	A human is omnivorous	A human has not stings	C	2
4	4	A giraffe is a person.	Giraffes can drink water from a lake.	A giraffe is not a human being.	.Giraffes usually eat leaves.	B	1

Fig. 9. Original Dataset for ComVE Subtask B

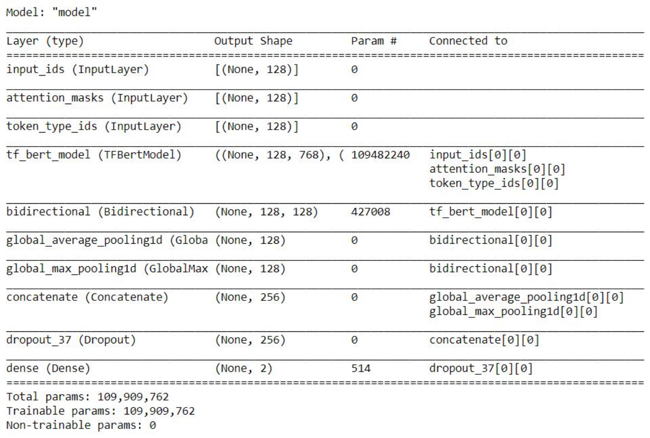
A sample of the transformed dataset for ComVE Subtask B is shown in Fig. 10. We have paired each false statement with each option for reasons in the original dataset and assigned the label accordingly. As shown in figure 10, we can see how one falsesent is paired with three of it's reason-options along with the class label. We transformed our training, dev and test datasets to this format and hence the size of our training, dev and test dataset is now 3*(original size).



	FalseSent	Option	Class
0	He poured orange juice on his cereal.	Orange juice is usually bright orange.	0
1	He poured orange juice on his cereal.	Orange juice doesn't taste good on cereal.	1
2	He poured orange juice on his cereal.	Orange juice is sticky if you spill it on the ...	0
3	He drinks apple.	Apple juice are very tasty and milk too	0
4	He drinks apple.	Apple can not be drunk	1
5	He drinks apple.	Apple cannot eat a human	0
6	Jeff ran 100,000 miles today	100,000 miles is way to long for one person to...	1
7	Jeff ran 100,000 miles today	Jeff is a four letter name and 100,000 has six...	0
8	Jeff ran 100,000 miles today	100,000 miles is longer than 100,000 km.	0

Fig. 10. Transformed Dataset for ComVE Subtask B

Post the transformation of the dataset to the sentence-pair format, we again used the BERTSemanticDataGenerator() to generate batches of data using the pretrained bert-based-uncased Tokenizer and model as described in Section VI - Subtask A using pretrained BERT model. We generated the batches of data and added trainable layers on top of frozen layers to adapt the pretrained features on the new data. We used the hybrid pooling to the bi-lstm sequence output and unfroze the model to recompile it with some hyperparameters. We finally used the trained model to make predictions on our test dataset.



```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
input_ids (InputLayer)         [(None, 128)]        0
__________________________________________________________________________________________________
attention_masks (InputLayer)   [(None, 128)]        0
__________________________________________________________________________________________________
token_type_ids (InputLayer)    [(None, 128)]        0
__________________________________________________________________________________________________
tf_bert_model (TFBertModel)    ((None, 128, 768), ( 109482240   input_ids[0][0]
                                                                attention_masks[0][0]
                                                                token_type_ids[0][0]
__________________________________________________________________________________________________
bidirectional (Bidirectional)  (None, 128, 128)     427008      tf_bert_model[0][0]
__________________________________________________________________________________________________
global_average_pooling1d (Globa (None, 128)         0           bidirectional[0][0]
__________________________________________________________________________________________________
global_max_pooling1d (GlobalMax (None, 128)         0           bidirectional[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)      (None, 256)          0           global_average_pooling1d[0][0]
                                                                global_max_pooling1d[0][0]
__________________________________________________________________________________________________
dropout_37 (Dropout)           (None, 256)          0           concatenate[0][0]
__________________________________________________________________________________________________
dense (Dense)                  (None, 2)            514         dropout_37[0][0]
==================================================================================================
Total params: 109,909,762
Trainable params: 109,909,762
Non-trainable params: 0
```

Fig. 11. Model Summary for Subtask B - using pretrained BERT model

We then wrote a custom Python module to transform our results to the desired format that could be read by the scorer function for subtask B. The function would compare the class 0 and class 1 prediction probabilities across all the three values of the FalseSent-OptionA, FalseSent-OptionB, FalseSent-OptionC for each Statement-Reason instance. Thereafter, we would return the class which has the highest probability of being the most appropriate reason for the Statement not making sense.

Using this approach, we were finally able to get the Class containing the most appropriate reason for the statement which does not make sense. We redirected this output to a CSV file that we would provide to the scorer function to evaluate the accuracy of our prediction.

C. Subtask C - Explanation (Generation)

The objective of Subtask C (Explanation Generation) is to train the machine to generate reasons which will be evaluated using BLEU scores as they will be assessed against the referential reasons. ***Task C: Explanation (Generation)***

Task C: Generate the reason why this statement is against common sense and we will use BLEU to evaluate it.

Statement: I like to ride my chocolate.

Referential Reasons:

1. Chocolate is a food, not a transportation unit.

2. You eat chocolate not ride it.

3. Can't ride chocolate.

For subtask C, since the task is to generate text, we had to use an autoregressive model like GPT2. We fine-tuned the GPT2 model to our custom dataset to generate the explanation (reasons) for the identified statement that was against commonsense. We used the GPT2 124M variant to fine tune our model as the variants higher than than (345M, etc) were too large to be trained on our system and we could see out-of-memory issues. We downloaded the fine-tuning repository and model [25]. We then encoded the data to translate the data from human readable text into machine language which the model understands. We encoded our .txt dataset (containing statement and referential reasons) of subtask C into NumPy's compressed array format .npz format. We then fine-tune the GPT2 model by training it on our custom dataset for around 13000+ epochs where we were able to substantially reduce the loss from 2.53 to 0.20. We stopped training further to avoid overfitting to the training dataset and use the validation loss to determine when to stop. Finally we used this model to generate reasons on the test dataset for subtask C.

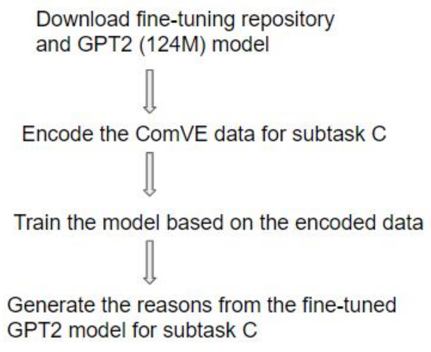
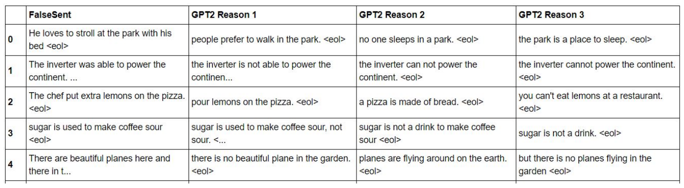

Download fine-tuning repository and GPT2 (124M) model
Encode the ComVE data for subtask C
Train the model based on the encoded data
Generate the reasons from the fine-tuned GPT2 model for subtask C

Fig. 12. Subtask C - Steps to fine-tune GPT2 model

As part of post-processing, we picked up all the reasons generated by the GPT-2 model on the test dataset and appended ¡eol¿ to the end of all the generated reasons to finally have reasons generated as shown in Fig. 13. Finally we used the get_close_matches() module from the difflib library to extract the most relevant one from the three reasons that were generated and moved it to a CSV file that would be passed to the scorer function for task C to find the BLEU score of our generated text.



	FalseSent	GPT2 Reason 1	GPT2 Reason 2	GPT2 Reason 3
0	He loves to stroll at the park with his bed <eol>	people prefer to walk in the park. <eol>	no one sleeps in a park. <eol>	the park is a place to sleep. <eol>
1	The inverter was able to power the continent. ...	the inverter is not able to power the continen...	the inverter can not power the continent. <eol>	the inverter cannot power the continent. <eol>
2	The chef put extra lemons on the pizza. <eol>	pour lemons on the pizza. <eol>	a pizza is made of bread. <eol>	you can't eat lemons at a restaurant. <eol>
3	sugar is used to make coffee sour <eol>	sugar is used to make coffee sour, not sour. <...	sugar is not a drink to make coffee sour <eol>	sugar is not a drink. <eol>
4	There are beautiful planes here and there in t...	there is no beautiful plane in the garden. <eol>	planes are flying around on the earth. <eol>	but there is no planes flying in the garden <eol>

Fig. 13. Subtask C - Reasons generated by fine-tuned GPT2 model

VII. Evaluation Strategy

Subtask A and B of the ComVE challenge was evaluated based on the ***accuracy*** of the prediction. Subtask C was evaluated using ***BLEU (Bilingual Evaluation Understudy) score*** which is calculated by comparing the generated reason against the referential reasons. It measures the difference between an automatic generated text and one or more human created referential text, which in our case are the referential reasons.

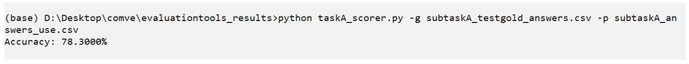
For evaluation of results, we were provided with scorer functions for all the three subtasks on GitHub [17]. All these scorer functions are .py files which require as argument the CSV file with the prediction results. The CSV file's format should be similar to the gold-answer's format that has been packaged with the data. We simply pass the CSV file to the scorer functions, to get the accuracy score for subtask A and subtask B and the BLEU score for subtask C.

VIII. Results Analysis

We passed our output CSV files to the scorer functions for each subtask and evaluated the performance of our model on the test dataset. The results for each subtask are highlighted below.

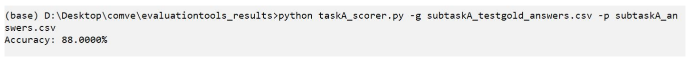
A. *Subtask A - Validation*

For Subtask A we followed 2 approaches - Google's USE and pre-trained BERT model approach. Fig. 14 and 15 show how we used the scorer function to evaluate our accuracy scores for this subtask. We were able to achieve an accuracy of 78.3% for the Google's USE model and 88% for the pretrained BERT model.



```
(base) D:\Desktop\comve\evaluationtools_results>python taskA_scorer.py -g subtaskA_testgold_answers.csv -p subtaskA_an
swers_use.csv
Accuracy: 78.3000%
```

Fig. 14. Subtask A (Google's USE): Accuracy obtained from scorer function

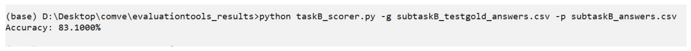


```
(base) D:\Desktop\comve\evaluationtools_results>python taskA_scorer.py -g subtaskA_testgold_answers.csv -p subtaskA_an
swers.csv
Accuracy: 88.0000%
```

Fig. 15. Subtask A (Pretrained BERT): Accuracy obtained from scorer function

B. *Subtask B - Explanation (Multi-Choice*

For Subtask B we used the pre-trained BERT model approach. Fig. 16 shows how we used the scorer function to evaluate our accuracy scores for this subtask. We were able to obtain an accuracy of 83.1% for subtask B.

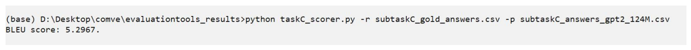


```
(base) D:\Desktop\comve\evaluationtools_results>python taskB_scorer.py -g subtaskB_testgold_answers.csv -p subtaskB_answers.csv
Accuracy: 83.1000%
```

Fig. 16. Subtask B (Pretrained BERT): Accuracy obtained from scorer function

C. *Subtask C - Explanation (Generation*

For subtask C, we fine-tuned the GPT2 124M model to generate the reasons. Fig. 17 shows how we used the score function to evaluate the BLEU score for the generated reasons. We were able to achieve a BLEU score of 5.2967.



```
(base) D:\Desktop\comve\evaluationtools_results>python taskC_scorer.py -r subtaskC_gold_answers.csv -p subtaskC_answers_gpt2_124M.csv
BLEU score: 5.2967.
```

Fig. 17. Subtask C (Fine-tuned GPT2) : BLEU score obtained from scorer function

We also evaluated our scores against the scores that were submitted as part of the results for the SemEval 2020 ComVE [26] to know our rank for each subtask. The summary of our results is shown in Table IV. The rank in the table shows the rank obtained by our models out of the total number of submissions.

TABLE IV
Summary of ComVE Results

Subtask	Model	Eval. Metric	Score	Rank
A	Google's USE	accuracy	78.3%	34/45
A	Pretrained BERT	accuracy	88%	23/45
B	Pretrained BERT	accuracy	83.1%	17/35
C	Fine-tuned GPT2	BLEU	5.2967	17/24

IX. Limitations

We would like to list out some of the factors that were challenging and limited the implementation of our project. Working on these factors in the future, could lead to more accurate and efficient results:

- For Subtask C, fine-tuning a GPT2 variant with a higher number of parameters (345M or 1.5B rather than 124M) could help us generate more accurate reasons that could match to the ones provided by humans, thereby leading to an increased BLEU score. Our system could only handle a maximum of the 124M variant for the GPT2 without running into OOM (out-of-memory) issues.
- Additionally, using GPT3 which is trained in 175B features and also supports few shot learning could be a very effective model for Subtask C as we could provide the sample reasons for the model to learn better, but GPT3 was not available open-source and it was impossible to train a model with 175B on our system.

- We feel that having more data for training and validation could have made much difference in the accuracy of predictions for subtask A and subtask B and also the GPT2 model would have been fine tuned even better. Currently we had 10000 training instances and 1000 validation instances, adding more data could help the models to learn and perform better.

X. Conclusion

With the advent of transfer learning, the ability of machines to understand Natural Language has enhanced significantly. The prime motive of taking up the ComVE challenge was to understand how challenging it is for machines to understand the text and identify sense in statements. Specific to this project, it was evident that transfer learning has been the way to go in order to introduce and identify commonsense in NLU tasks. The fact that pretrained NLU models like BERT, GPT2, GPT3 have been trained on a massive amount of textual data and have features as high as 175B, explains how intelligent these models are and how we can leverage these models using transfer learning to accomplish various natural language understanding tasks like ComVE. As part of the ComVE challenge, we constructed a total of four models. Two models for subtask A, one for subtask B, and another one for subtask C. For Subtask A, we used Google's Universal Sentence Encoder to encode sentences and classify them based on whether they make sense or not, and we also used the pretrained BERT model for the same. To the pretrained bert model and tokens, we added the Bi-LSTM layer to adapt the pretrained features on the custom ComVE dataset and used the hybrid pooling approach on the bi-lstm sequence output. For Subtask A, we were able to achieve an accuracy of 78% for USE and 88% for BERT. For Subtask B, we used the pretrained BERT model as we passed the false statement and reason(s) as pairs and processed the output to extract the most relevant reason with an accuracy of 83%. For Subtask C, we fine-tuned GPT2 124M model to our custom dataset and we could achieve a BLEU score of 5.297. This work was a great learning experience as we were exposed to a lot of recent NLU technologies and the essence of transfer learning to achieve exceptional results in the tasks. We got an opportunity to work with transformers, an auto regressive model (GPT2), Google's sentence encoders, use GPU for faster training and also learn about various CUDA libraries available in Keras that enables faster model training. Commonsense validation is one of the newest and unexplored areas of NLP, and it was paramount to understand how we could leverage the various available NLU technologies to achieve results. The experiments listed out as the future work for this project will be a significant direction to achieve better performance and insights about Commonsense Validation and Explanation.

XI. Future Work

We would like to extend our work in the ComVE challenge by considering the below directions for future work and exploring these methods to see if we could further improve the accuracy and performance of our system.

- For sentence embeddings, we have tried the USE and BERT Tokenizer. It would be interesting to try some other sentence embeddings like Facebook's Sentence Embedding, InferSent etc to see how the model performs.
- Google's Multilingual Universal Sentence Encoder is trained on a larger number of features, so translating sentences using MUSE, could get some better results. However, for our implementation we did not use MUSE because the sentences were in English and not multilingual, but it would be insightful to see how the performance of the model is with MUSE.
- We have used the Bi-LSTM + bert-based-encased model with Hybrid Pooling. Hybrid pooling considers Average-Pooling (mean tokens) and MaxPooling (max tokens). However, we could try standalone pooling techniques based on mean tokens, cls tokens, max tokens only rather than using the ensemble (hybrid) approach and see how the model performs.
- We could also use other pretrained BERT variants - RoBERTa, DistilBERT to carry out subtasks A and B of the ComVE challenge.
- Instead of using a Bi-LSTM with the pretrained BERT model, we could also use a Multi layer perceptron to see how the model performs for subtask A and B. For subtask C, we have used the 124M variant of the GPT2 model. We could use the higher variants of GPT2 - 345M or 1.5 B for generation of text and carry out the experiment in a lab-station where the system could support large models without memory issues.
- We could raise a request via the GPT3 form to ask for access and fine tune the GPT3 model which supports few-shot learning and is also trained on 175B features to get more accurate results corresponding to human reasons for subtask C.

Acknowledgement

We would like to express our sincere gratitude to Professor Diana Inkpen for her valuable guidance and for providing us the necessary background on Natural Language Processing that helped us put our knowledge to practice for the implementation of this project.

References

[1] Competition. (n.d.). Retrieved January 13, 2021, from https://competitions.codalab.org/competitions/21080

[2] Davis, E., amp; Morgenstern, L. (2004). Introduction: Progress in formal commonsense reasoning. Artificial Intelligence, 153(1-2), 1-12. doi:10.1016/j.artint.2003.09.001

[3] [36] D.A. Randell, Z. Cui, A.G. Cohn, A spatial logic based on regions and connection, in: B. Nebel, C. Rich, W.R. Swartout (Eds.), Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR1992), Morgan Kaufmann, San Francisco, CA, 1992, pp. 165–176.

[4] P.R. Cohen, H.J. Levesque, Intention is choice with commitment, Artificial Intelligence 42 (2–3) (1990) 263–309.

[5] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 3320–3328.

[6] 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 Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[8] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In Proceedings of ACL, pages 1756–1765.

[9] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In 2011 AAAI Spring Symposium Series.

[10] Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. Transactions of the Association for Computational Linguistics, 5:379–395.

[11] 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.

[12] 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.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

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

[15] Ernest Davis. 2016. How to write science questions that are easy for people and hard for computers. AI magazine, 37(1):13–22.

[16] Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. arXiv preprint arXiv:1909.00277.

[17] Wangcunxiang. (n.d.). Wangcunxiang/semeval2020-task4-commonsense-validation-and-explanation. Retrieved April 04, 2021, from https://github.com/wangcunxiang/SemEval2020-Task4-Commonsense-Validation-and-Explanation

[18] CunxiangWang, 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.

[19] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil. Universal Sentence Encoder. arXiv:1803.11175, 2018.

[20] TensorFlow hub. (n.d.). Retrieved April 07, 2021, from https://tfhub.dev/google/universal-sentence-encoder-large/1

[21] Transformers. (n.d.). Retrieved April 08, 2021, from https://huggingface.co/transformers/

[22] Bert-base-uncased · Hugging Face. (n.d.). Retrieved from https://huggingface.co/bert-base-uncased

[23] Openai. (n.d.). Openai/gpt-2. Retrieved from https://github.com/openai/gpt-2

[24] Alammar, J. (n.d.). The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Retrieved from http://jalammar.github.io/illustrated-bert/

[25] Nshepperd. (n.d.). Nshepperd/gpt-2. Retrieved from https://github.com/nshepperd/gpt-2

[26] Competition. (n.d.). Retrieved from https://competitions.codalab.org/competitions/21080results

[27] Reimers, N., Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). doi:10.18653/v1/d19-1410

[28] Zellers, R., Bisk, Y., Schwartz, R., Choi, Y. (2018). SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. doi:10.18653/v1/d18-1009

[29] Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., . . . Allen, J. (2016). A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. doi:10.18653/v1/n16-1098

[30] Zhang, H., Zhao, X., Song, Y. (2020). WinoWhy: A Deep Diagnosis of Essential Commonsense Knowledge for Answering Winograd Schema Challenge. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.508