Story Cloze Test with Fine Adjustment Schemes of GPT-2 Model

Xizhe Wang, Zinan Xiong
Department of Computer Science
Kennedy College of Sciences
University of Massachusetts Lowell
Lowell, Massachusetts
United States, 01854
{xizhe_wang, zinan_xiong}@student.uml.edu

***Abstract*—Natural Language Processing (NLP) is one of the most popular and crucial fields in deep learning research. The applications of NLP can vary from speech recognition, natural language understanding, language generation, and so forth. *Story Cloze Test* is an evaluating narrative structure learning task that replaces its predecessor task Narrative Cloze Test. In the Story Cloze Test, there are a new corpus of five-sentence stories, which is called 'ROCStories'. The corpus of this task is unique in two aspects: 1) it captures a rich set of causal and temporal commonsense relations between daily events, and 2) it is a high quality collection of everyday life stories that can also be used for story generation. There are many works tried to solve this problem. Amongst all them, the generative pre-training II (GPT-2) model is one of the newest and most efficient word, and to our best knowledge, can achieve state-of-the-art performance on the task of *Story Cloze Test*. Therefore, GPT-2 is chosen as our fundamental model in this paper. Moreover, we propose with two schemes, namely sentence trunk scheme and antonym sentences generating, to try to make some further improvements for GPT-2 model. Our proposed schemes perform as much accuracy as the original GPT-2 model. Though our proposed schemes did not make much improvements, they still prove that our general ideas are valuable to be further studied and might offer a promising direction in this task.**

***Index Terms*—GPT-2, part-of-speech tag, sentence trunk, antonym sentence**

I. Introduction

Story completion is always the ambitions in artificial intelligence. One of the challenges in expanding the field had been the lack of a solid evaluation framework and datasets on which comprehension models can be trained and tested.

Story Cloze Test [7] was introduced by Mostafazadeh et al. (2016) to address this issue. The task gives a four-sentence short story as the 'context', and should choose the correct ending sentence from two options. It requires combining semantic understanding and common-sense knowledge of our world [5].

The main issue of this task is, the training set only has one ending, so we have to find a way to generate the second ending, otherwise, we can only use the validation set for training.

Though some people claimed that they achieved a good result even without using training dataset, we think it will benefit our model if we can augment our training set, and utilize all the stories in training dataset. So one of our contribution will be augment the dataset, and achieve a reasonable result which is not far from the result we get from the model trained only with validation dataset.

Several shallow and state-of-the-art models were tried to tackle this task, however they only show a slightly better result than randomly guessing [6].

The Transformer based models are very popular today since they have shown very powerful ability in the field of natural language processing and can achieve a very good result in different NLP tasks, so we are going to choose one of them as our model. The model we are going to use is based on GPT-2 small, which is a decoder only structure with masked self-attention mechanism, consists of 12 layers, and has 117M parameters.

Recurrent Neural Networks suffer from short-term memory problem, and if a sequence is long enough, it will have a hard time carrying information from earlier time steps to later ones. LSTM 's and GRU's were created as the solution to short-term memory [11]. They have internal mechanisms called gates that can regulate the flow of information. The GRU [14] is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU's got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate.

A sequence-to-sequence model [15] is a model that takes a sequence of items (words, letters, features of an images…etc) and outputs another sequence of items [8]. The sequence-to-sequence model is composed of an encoder and a decoder, where the encoder processes each item in the input sequence, compiles the information it captures into a vector, and then pass the final context vector to the decoder. The decoder will then produce the output sequence item by item and generate the whole sentence. In this model, the encoder and decoder tend to both be RNNs.

Transformer is a model that uses attention to boost the training speed, it consists of one encoding component, one decoding component, and a connection between them [1]. The encoding component is a stack of same encoders, and

the decoding component is a stack of decoders of the same number. The structure of the transformer is shown in Figure 1, and it is the core component of a lot of transformer based structures, including BERT and GPT-2.

II. Related Works

A. Other NLP Works

The release of BERT is described as marking the beginning of a new era in NLP. BERT builds on top of several clever ideas including Semi-supervised Sequence Learning, ELMo, ULMFiT, and Transformer [2]. BERT is a trained Transformer Encoder stack (12 for the Base version and 24 for the Large version), it has large feedforward-networks and more attention heads in the Transformer. BERT takes a sequence of words as input which keep flowing up the stack, each layer applies self-attention and passes its results through a feed-forward network, then it will hands them over to the next encoder.

Part-of-Speech [12] is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context. It is useful in information retrieval, text to speed, and word sense disambiguation.

Semantic Role Labeling (SRL) [9] is used to classify arguments of predicates into a set of participant types or semantic roles, like 'agent', 'patient', 'instrument', 'beneficiary', and 'source'. It aims to recover the verb predicate-argument structure of a sentence, describes the semantic relation between the arguments of the verb and the situation described by the verb, such as who did what to whom, when, why where and how.

Named entity recognition (NER) [13] seeks to locate and classify named entities in text into pre-defined categories such as the names of persions, orgnizations, locations, expressions of times.

AllenNLP [3] offers a state of the art SRL tagger and NER tool that can be used to map semantic relations between verbal predicates and arguments.

Antonym generation is a tough task in Natural Language Processing, because in a lot of language models, the word embedding of the original word and its antonym are usually pretty close, and the same word may have different antonyms under different situation. WordNet [4]is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. The structure of WordNet makes it a useful tool for computational linguistics and natural language processing, and it is also useful to find the antonym of specific word.

B. Dataset Study

There are totally 6 csv files provided by ROCstory, two training datasets containing 98161 stories in total, two validation datasets containing 1571 and 1886 stories respectively, and two test datasets containing 1871 stories respectively. The first four sentences are given for training, and the last sentence is the one need to be predicted and chose from, as shown in Table I. As per analysis, there are some duplicated stories among two validation datasets, so it need to be cleaned before it can be used.

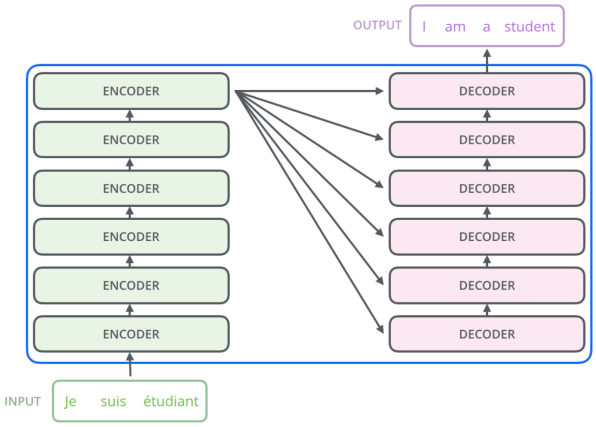

Fig. 1. The Structure of Transformer.

TABLE I
A SAMPLE STORY FROM OUR VALIDATION DATASET

Karen was assigned a roommate her first year of college.
Her roommate asked her to go to a nearby city for a concert.
Karen agreed happily.
The show was absolutely exhilarating.
Right Ending: Karen became good friends with her roommate.
Wrong Ending: Karen hated her roommate.

III. GPT-2 Model

GPT-2 is a transformer-based model [10]. The original GPT-2 model is used to predict the next word given previous words or sequences. It is widely used in text generation applications. There four language models for GPT-2 totally as Figure 2 shows.

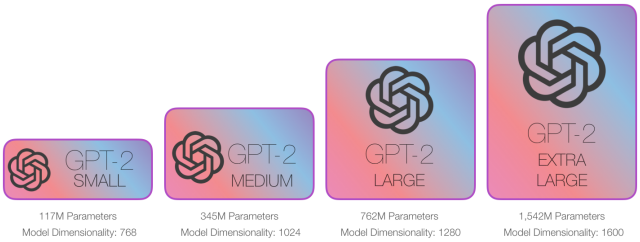

Fig. 2. Four Language Models.

The model we use in this paper is the smallest one, which contains approximately 117 megabytes' parameters. Its word embedding is in size of 768. As we choose the smallest one, the layer number of decoder block is 12 in default as shown in Figure 3. With different language models, we actually can have different sizes of GPT-2 models. Generally, the embedding size and decoder layer number will get larger as Figure 4 shown.

Different from other models, GPT-2 models only contain decoder blocks, while BERT-based models contain encoder blocks and sequence-to-sequence models contain both.

Fig. 3. Small Language Model of GPT-2.

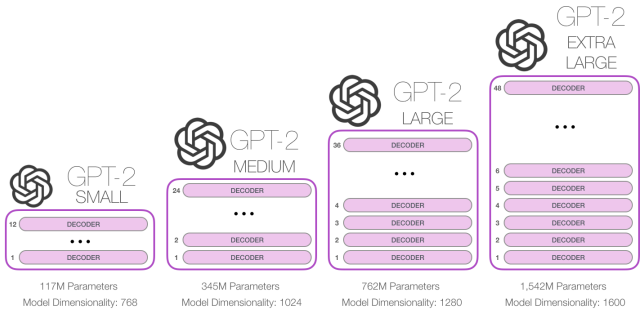

Fig. 4. Layer Numbers of Decoder Block of GPT-2 Models.

A. Decoder Block of GPT-2

This sub-section will introduce the details of the decoder block of GPT-2. There are two layers in each decoder block. The structure of a typical decoder block is shown in Figure 5 [16]. One layer is masked self-attention. The other one is a vanilla multi-layer perceptron.

Individually, a GPT-2 model processes each sentence token by token in the order within the sequence. With respect to a specific token, the input token should be processed into an input vector. Then, the input vector will be forwarded into the masked self-attention layer. At this step, we will get a z vector as a medium result. Further, the z vector will be forwarded to the multi-layer perceptron. At last of this decoder block, we will get an output vector.

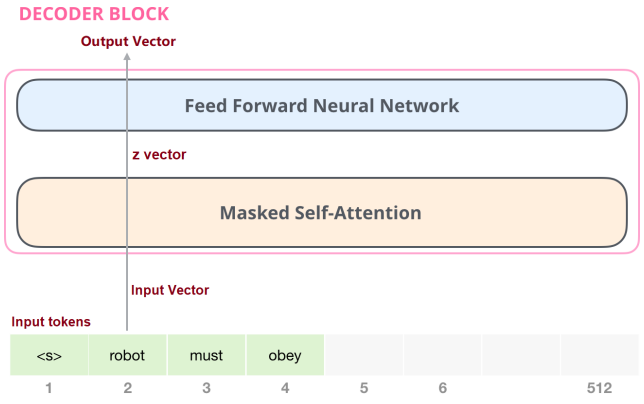

Fig. 5. Decoder Block Structure of GPT-2.

The operation mentioned by last paragraph is the entire procedure for only one decoder block. In terms of the whole GPT-2 model, the output vector generated by previous decoder block will be forwarded to the current decoder block. The current decoder block will process take the output vector as its input vector. After such operation, the output vector of current decoder block will be forwarded to the next decoder block. This process will be continuous until the last decoder block.

B. Positional Encoding

As previous sub-section mentioned, there is a procedure required to change the token embedding into a input vector. Generally, the procedure sums up up the token embedding and positional encoding for each token. The sum up procedure is shown in Figure 6.

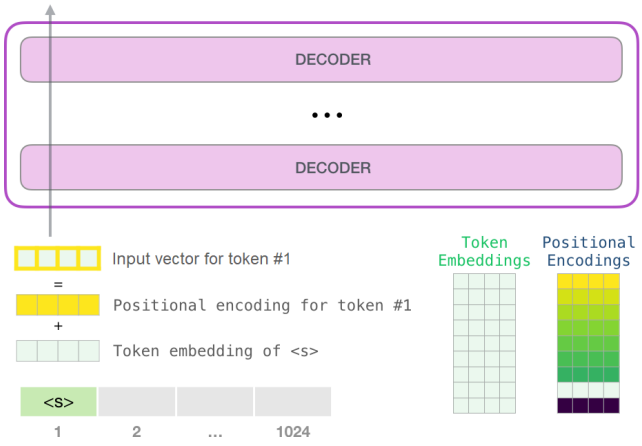

Fig. 6. Procedure of Positional Encoding Sum-up.

The formulas (1) bellow show how the positional encoding is calculated.

$$\begin{aligned} \mathrm{PE}(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/d_{\text{modd}}}}\right), \\ \mathrm{PE}(pos, 2i+1) &= \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \end{aligned} \tag{1}$$

where pos refers to the position index of the specified token in the sequence; d_{model} represents the size of token embedding, which is 768 in this task; $2i$ and $2i+1$ are the even number index and odd number index of the scalar corresponding to the positional encoding vector. The formulas explained how every scalar in the positional encoding vector is calculated.

The main purpose of the positional encoding vectors are to add the position information into each token in sequences. Because in GPT-2, there is no hidden state generated any more. So the positional encoding vectors are the only parameters to carrier over the position and sequence information.

C. Masked Self-Attention

The masked self-attention is the most important component in a GPT-2 model. The main purpose of the masked self-attention layer is to generate a group of scores that represent the connection degrees between the current focused token and all the tokens within the attention.

Firstly, this paragraph offers a explanation of the use of the value vectors and the scores. Basically, it just needs to sum up

all the value vectors in weighted average based on the scores as their weights. As shown in Figure 7, when it comes to the word "it" in the sequence of "a robot must obey the orders given it", the word "it" gets high scores with the words "a" and "robot". Intuitively, the pronoun "it" is, indeed, used to refer to "a robot" as the mean of this sentence.

Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

Fig. 7. Sum-up of Value Vectors with Scores.

In this paragraph, we will explain how to calculate the scores in detail. Firstly, there are three square matrices created, namely Query (or Q), Key (or K), Value (or V). An illustration of these three matrices is shown in Figure 8. All the three square matrices at this step, Q, K, and V, are 2-dimension and in size of 768 by 768, where 768 is corresponding to the vector size of token embedding in the small version of GPT-2 language model.

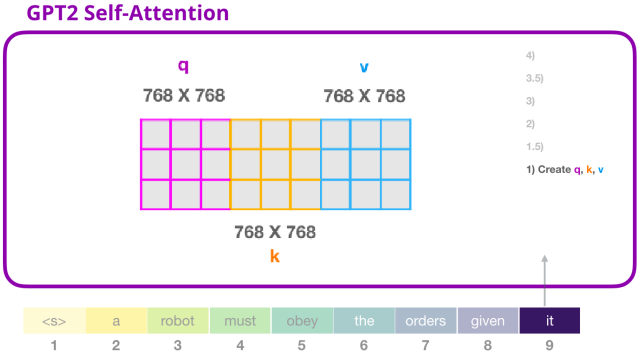

Fig. 8. Three Matrices of Q, K, and V.

Then, a multiplication, which is a cross product, of the input vector and the Q, K, and V matrices is implemented as shown in Figure 9. After the multiplication, the original matrices Q, K, and V become three vectors, namely q, k, and v. They are all in size of 1 by 768 as vectors. The equation is shown as formula (2):

$$(q, k, v) = I \times (Q, K, V), \tag{2}$$

where I refers to the input vector.

The v vector obtained from the last step is exactly the value vector we required to do the weighted average sum up to calculate the z vector. Thus, the only missing parameters

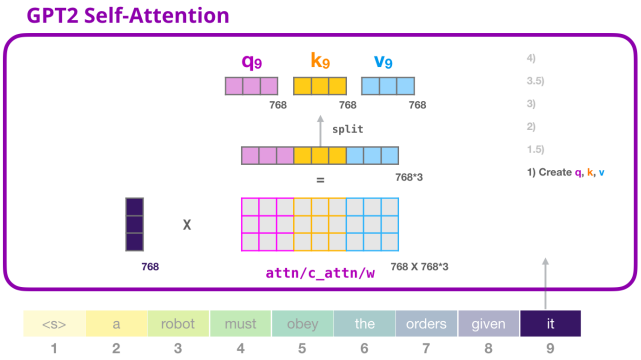

Fig. 9. Multiplication of Input Vector and Q, K, and V.

remained are the group of scores for the z vector calculation as the weights.

Then, we get a medium score of the current token and a specific token in the sequence by conducting dot product of the q vector of the current focused token and the k vector of all the previous masked self-attention tokens. The equation is shown in (3):

$$score'_i = q_{curr} \cdot k_i, \tag{3}$$

where $curr$ refers to the index of the current focused token. For example, the sequnce is "a robot must obey the orders given it", $curr$ = 8 if we focus on the token "it". i represents the index of previous tokens in this sequence that are masked as attention in current step. In this case, i can be an integer that starts from 0 to $curr$. For example, if i = 2, we will get the $score'_2 = q_8 \cdot k_2$ as a medium score between token "it" and "robot".

$score'_i$ is a scalar after all, and it is just a medium result rather than the score we required for the z vector calculation. Hence we put a prime, "$'$", for the $score'_i$. It needs to be further processed for the purpose of getting the final score we required. In a sequence perspective, we get the medium scores by the process shown in Figure 10. Then we apply attention mask to remove the scores of tokens that are supposed to block by the mask. The process is shown in Figure 11. Further, we carry out the softmax function to re-calculate all the scores. The process is shown in Figure 12.

At last of this masked self-attention layer, we will get the z vectors as we have obtained the v vectors and the group of corresponding scores $score_i$. The equation is (4):

$$z_{curr} = \sum_{i=0}^{curr} socre_i \cdot v_i, \tag{4}$$

where z is a vector in the same size as v vectors. $curr$ refers to the index of the current focused token.

D. Multi-Head Attention

In GPT-2, there is a technique, which is defined as multi-head attention, used as a crucial aspect during the masked self-attention process. The general multi-head attention process is shown in Figure 13 and 14. Typically, the multi-head attention

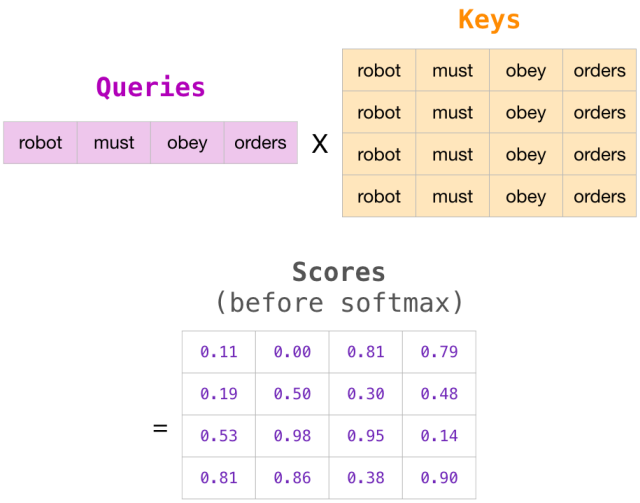

Fig. 10. A Sequence Perspective of Getting the $score_i'$ Values.

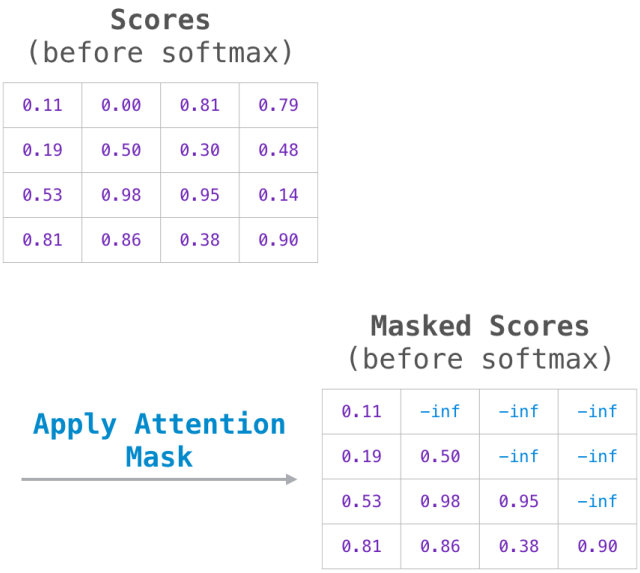

Fig. 11. Matrix $score_i'$ after Attention Block.

algorithm of GPT-2 splits the q vector into 12 equally divided vectors, which are all in size of 1 by 64 as the size of the original q vector is 1 by 768. At the same time, the k vector and v vector are going through the same splitting process.

The q, k, and v vectors are splitted into 12 parts, which are defined as 'heads'" in the GPT-2 model. Then, we conduct the same process as equation (4) in the 12 different heads, individually. The typical procedure is shown in Figure 15. As the q, k, and v vectors are in a smaller size, the z Vectors will also be in a smaller size. Therefore, we just need to concatenate the splitted z vectors at last to recover it back to its original size.

The multi-head attention technique helps the GPT-2 model to capture various aspects of the input information and improve the expressive ability.

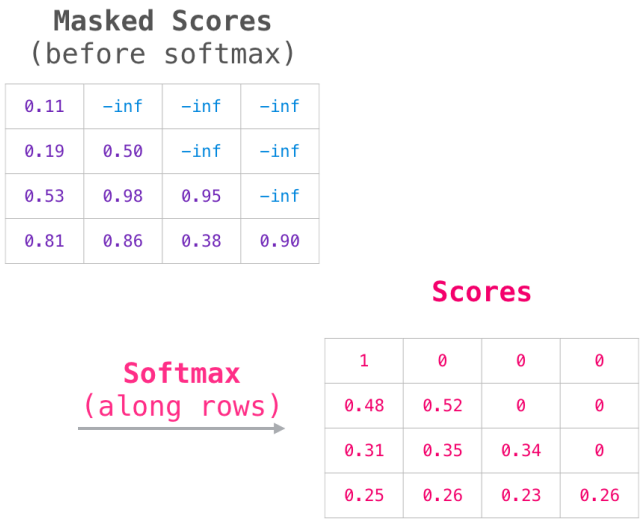

Fig. 12. Final Matrix $score$.

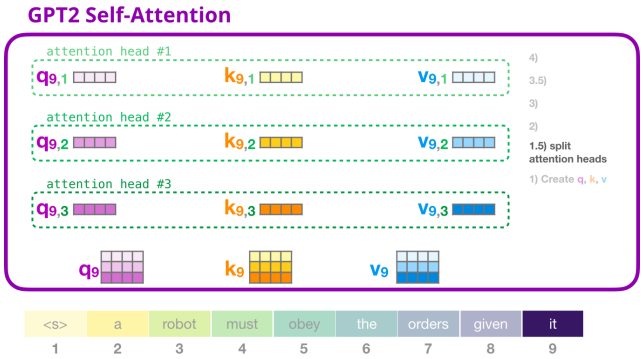

Fig. 13. Multi-Head Attention 1.

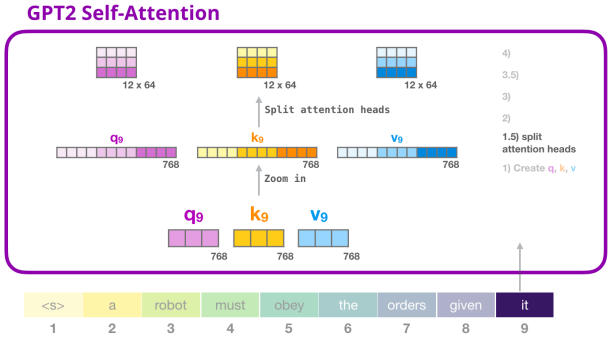

Fig. 14. Multi-Head Attention 2.

E. Multi-Layer Perceptron

The z vector we obtained from the masked self-attention will be forwarded into the Multi-Layer Perceptron (MLP). The MLP here is just a vanilla neural network. The illustration of this process is shown in Figure 16. Finally, the MLP will return a output vector, which is in size of 1 by 768, the same as the size of the input vector and z vector.

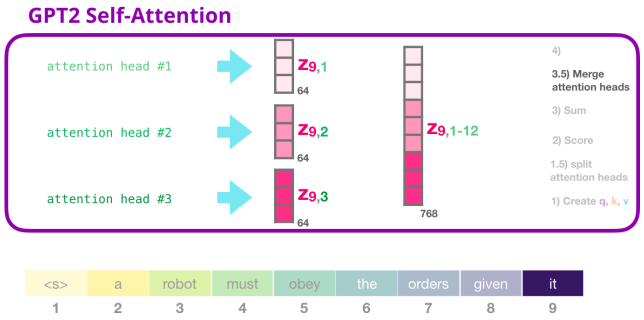

Fig. 15. Concatenation of the z Vector.

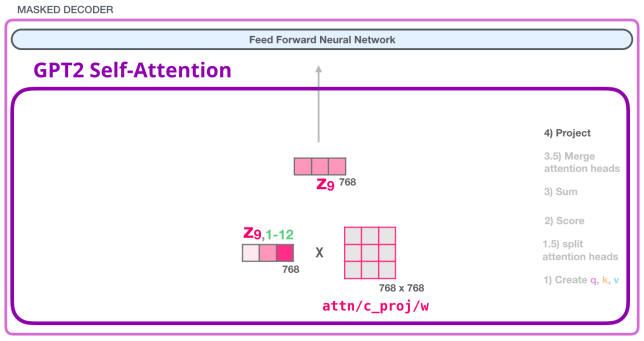

Fig. 16. Multi-Layer Perceptron.

F. Adjustments for Story Cloze Test

The original version of GPT-2 model is used to predict words or sequences, a.k.a. the text generation, which is a regression task. The Story Cloze Test is aimed at choosing the correct story endings, which is a classification task. As a result, the GPT-2 model should be changed in a degree to apply to the purpose of the new task.

In the end of each sequence, as a typical case is shown in Figure 17, it adds a special token called <classification>. Then, we will get an output vector for this token. The output vector of this special token is used to do the classification. We actually do not use the output vectors of previous sequence at all. The whole training process is focused on training the special token's output vector. When we get the output vector of this <classification> token, we simply do cross entropy the get its category.

IV. Experiments of Syntactic and Semantic Schemes

We propose two different schemes to try to make some improvements for the GPT-2 model applying to the Story Cloze Test. One is defined as sentence trunk scheme and the other is defined as antonym sentences generating. These two shcemes will be described in details in the following sub-sections.

A. Sentence Trunk Scheme

The structure of some sentences in our training set is complicated, with multiple verbs or adjectives. An intuitive way to deal with this is to simplify the training data, by

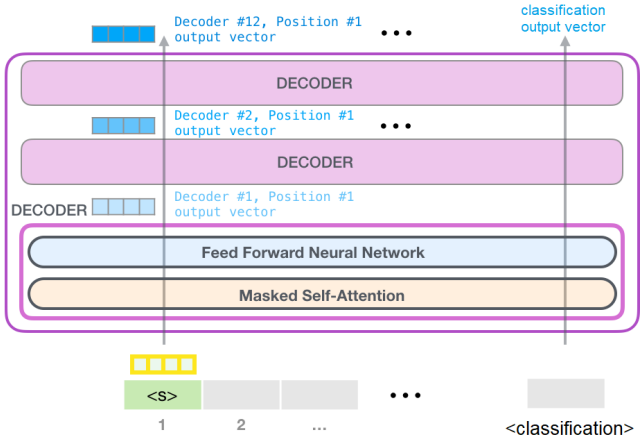

Fig. 17. GPT-2 Model for Story Cloze Test.

only extract the main components of each sentence. We can extract the subject, verb, noun, adjective, and adverb from the sentence, and discard all other components that won't affect the meaning of our sentence. For example, 'The dog ate the apple' is transformed to 'Dog ate apple', only the main components are preserved. Another example is 'Rick grew up in a troubled household' is transformed to 'Rick grew up troubled household', so we can extract the verb and adjective later.

AllenNLP embed each word in a low-dimensional space, pass them through an LSTM to get a sequence of encodings, and use a feedforward layer to transform those into a sequence of logits (corresponding to the possible part-of-speech tags. So when we send 'Rick grew up in a troubled household', it will return a structured sentence with POS tagging like '(S (NP (NNP Rick)) (VP (VBD grew) (PRT (RP up)) (PP (IN in) (NP (DT a) (JJ troubled) (NN household)))) (. .))', where the corresponding tags are listed in Table II.

We try to modify the GPT-2 model to imitate the approach how human beings understand sentences or stories. In an intuition, people extract the trunks of the sentences or stories even though reading of all the words for contents. Under such principles, we only keep the truck of each sentence in every story. The trunk we defined in our trials are noun (NN), verb (VB), adjective (JJ), adverb (RB), conjunction (CC), and punctuation. They are regarded as the "important" parts of the sentences. Moreover, we remove all the other parts in the sentences as they are relatively not important. For instance, if the original sentence is "I tried going to the park the other day.", it will be extracted to "I tried going park other day.".

We extracted and translated all the sentences of all the stories for the validation dataset and testing dataset. All the experiments show that our general concepts are feasible and can be further studied in the future researches. The evaluation will be listed in the next section.

B. Antonym Sentences Generating

Antonym generation is a tough task in natural language processing, since in most of the language models, the word

TABLE II
POS TAGGING ANNOTATION

S	Subject
NP, NNP	Proper none, singular
VBD	Verb, past tense
RP	Partical
JJ	Adjective
DT	Determiner
NN	None, singular or mass
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative

embedding for the original word and its antonym are quite close to each other, because they often appear together with each other. For example, 'good' and 'bad' have a very high similarity after word embedding with GloVe or word2vec.

In order to find the antonym, we used WordNet developed by Princeton University. WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. In WordNet, adjectives are organized in terms of antonymy, it consists of pair of words like 'wet-dry', and 'young-old', which reflects the strong semantic contract of their members.

To generate the antonym of a sentence, we defined a logic, that we first extracted the verb, adjective and adverb from the sentence with POS tagging, and set a counter. If the verb is consist of "n't", for example, "did n't", then we will remove the "n't" immediately and won't touch the counter. However, if there is no "n't" in the sentence, then we will search for the antonym of the adjective and adverb extracted in WordNet. If the antonym exists, we'll add one to the counter, and the same for the verb. In order to make sure the sentence has the opposite meaning, we will only change odd number of words we found. For example, if the list we found is [V, ADJ, ADJ, ADV], then we will add a 'not' in front of V, and get the antonym of the second ADJ, then put them back to the original sentence.

This is not the best logic, but we think it will at least change the meaning of the target sentence in training set and augment our training data, then help us training the model.

V. PERFORMANCE EVALUATION

We trained our model based on four different datasets we pre-processed, and then evaluated them on the same test set which contains 1871 stories.

In baseline model, we used the original validation set as our training data. In case 1, we used reorganized validation set as training data, and tested on reorganized testing set. In case 2, we used reorganized validation set as training data, and tested on the original testing set. In case 3, we combined the manually generated antonym sentences with the validation set as our new training set, and then tested on the original test set. In case 4, we only used the last sentence of the validation set as our training data, and tested on the original testing set.

TABLE III
EVALUATION RESULT

	Baseline	Case 1	Case 2	Case 3	Case 4
validation	90.64%	89.04%	90.91%	98.38%	87%
test	86.53%	86.37%	85.89%	85.36%	84.5%

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

From the result we can see, unfortunately, though all the results are quite close, none of them surpassed the result of the baseline model, even trained a much larger training set with manually generated wrong endings (case 3).

We can tell from the result, the method we used to generate the antonym sentences is not that good, since we manually added a 'not' in front of most of the verbs and set it as the wrong ending, the model easily learnt that and made predictions utilizing it.

An interesting thing is that even trained with only the last sentence of the validation data, the model still achieved a fairly good result, only about 2 percent lower than the baseline, which proved what we read from some paper, that the last sentence of the story plays a very important role in the prediction.

B. Future Work

The next step of our work will include the following.

1) Find a better structure to extract those main components from the sentences, so we will not lose some important information and get different meanings as the original sentence.
2) Integrate the remove operations into masked self-attention.
3) Since the antonym sentences may play an important role if we can fully utilize them, we will figure out a better method to generate the antonym sentences, instead of using WordNet to get it manually.
4) From the result in Table III, we can see it might have over-fitting issue. So after we fixed the antonym sentence issue, we should test and see if over-fitting is fixed. If not, then we need to look into it, and find a way to resolve this issue.
5) Once the over-fitting issue if resolved, we will go and try to implement the more powerful GPT-2 medium or GPT-large model, see if it will give a even better result.
6) The model we constructed now can only utilize one single GPU, which makes the training slow and take days. We'll try to implement multi-GPU in our model, so the training would take less time.

VII. ACKNOWLEDGEMENT

This paper is based on our NLP course project. We are very grateful for the hard working on teaching the NLP course of Professor Anna Rumshisky. She ushers us in the NLP field, where we feel full of interesting now. All the models and algorithms she introduced are helpful to our machine

learning/deep learning research. Also, we are grateful for all the teaching assistants' hard working during this semester. We would never learn so much without all the TAs' patience and supports.

REFERENCES

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[2] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[3] Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M. and Zettlemoyer, L., 2018. Allennlp: A deep semantic natural language processing platform. arXiv preprint arXiv:1803.07640.

[4] Miller, G.A., 1998. WordNet: An electronic lexical database. MIT press.

[5] Srinivasan, S., Arora, R. and Riedl, M., 2018. A simple and effective approach to the story cloze test. arXiv preprint arXiv:1803.05547.

[6] Lin, C.Y. and Hovy, E., 2002, July. Manual and automatic evaluation of summaries. In Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4 (pp. 45-51). Association for Computational Linguistics.

[7] Cai, Z., Tu, L. and Gimpel, K., 2017, July. Pay attention to the ending: Strong neural baselines for the roc story cloze task. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 616-622).

[8] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[9] Palmer, M., Gildea, D. and Xue, N., 2010. Semantic role labeling. Synthesis Lectures on Human Language Technologies, 3(1), pp.1-103.

[10] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I., 2018. Improving language understanding by generative pre-training. URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.

[11] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[12] Ratnaparkhi, A., 1996. A maximum entropy model for part-of-speech tagging. In Conference on Empirical Methods in Natural Language Processing.

[13] Sang, E.F. and De Meulder, F., 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. arXiv preprint cs/0306050.

[14] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[15] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[16] Alammar, J., The Illustrated GPT-2 (Visualizing Transformer Language Models). The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time. Available at: http://jalammar.github.io/illustrated-gpt2/ [Accessed May 1, 2020].