Leverage Auto-generation for Commonsense Explanation

Yueyi Wang **Xinyu Wang** **Yanjian Zhang**
School of Data Science
Fudan University
{16307130287, 16307130286, 16300200020}@fudan.edu.cn

Abstract

Inspired by previous work using explanation for CommonsenseQA(Rajani et al.), we conduct experiments on enhancing the Commonsense Explanation with auto-generated explanation with UniLM. And we achieve 0.7% increasement compared to our baseline accuracy of 78.71%. We further explore the likelihood to improve our performance by capturing inner semantic information with OpenIE and enhancing the generator with Adversarial training. A basic RNN-based adversarial experiment shows that it is promising to improve commonsense generation with adversarial training, which could in turn help our commonsense explanation task.

1 Introduction

Commonsense Reasoning has received increasing researcher's attention in recent years(Wang et al., 2019). With the development of the algorithm and the dataset, the machine can perform well on many benchmarks. But the intrepretability in commonsense reasoning is still a difficulty in Natural Language Processing(NLP). Embedded explanations is a way to verbalize the reasoning that the models learn during training(Rajani et al.). Commonsense Question Answering(CQA) is a multiple-choice question answering dataset proposed for developing NLP models with commonsense reasoning capabilities(Talmor et al., 2018). Generating explanations for commonsense reasoning can help a classifier model to make predictions(Rajani et al.).

Explanation	
Statement:	Dieting is a way to gain weight
Option1:	Dieting can change body shape
Option2:	If you eat less, you will lose weight
Option3:	Dieting is unhealthy

Reference	
Statement:	Dieting is a way to gain weight
Ref 1:	If you eat less, you will lose weight
Ref 2:	You generally lose weight when dieting.
Ref 3:	Dieting would help someone lose weight.

Table 1: Examples from dataset

As is shown in Table 1, in our dataset, given a statement with three options, the explanation task is to find the key reason why a given statement does not make sense. Correspondingly, we also have the reference information in generation task, three correct reasons of the wrong statement.

The researchers who provided the dataset have conducted experiments on `ELMo` and `BERT` to this task(Wang et al., 2019). We use their best presented model as a baseline for our work.

We fine-tune BERT with task of purely predicting the explanation option with statement. Next, based on BERT, We want to improve the performance by adding extra information to the dataset.

We have tried two methods to extract the information in our task. First, we extract "subject-relation-object" triples from the statements and options with Standford OpenIE[1], and add them to the dataset to train the model to enhance inner semantic representation. Second, we apply the generation model for auto-generated explanation to this task.

We make a trial on enhancing the commonsense explanation through generating an explanation for the wrong statement by using the reference information with UniLM(Dong et al., 2019). Then, we introduce GAN to enhance generation. But because of the time limitation, we only prove that GAN can indeed improve the performance of generation with RNN-based version, and leave the UniLM-based version as future work.

We aim at combining explanation and generation together and forming a adversarial architecture with BERT and UniLM. In this dataset, the result for BERT is 78.71%, the result for BERT+Generation with feature concatenation is 79.11%, and BERT+Generation with textual joining is 79.41%. More detailed information, such as the model, result analysis, will be shown in following chapters.

The contribution of our project are threefold:

- We append the explanations generated by current state-of-the-art model UniLM as extra information and achieve better performance in Commonsense Explanation task.
- We deliver comparable experiment results to show the effect of inner semantic information extracted from OpenIE.
- Our attempt on applying adversarial training proves to be useful for generation with simple experiment, which would in turn help commonsense explanation.

[1]https://nlp.stanford.edu/software/openie.html

2 Related Work

Commonsense Reasoning Commonsense reasoning has recently drawn significant attention. It is an interesting and challenging task, because inference requires information beyond input data. There are several subtasks under commonsense reasoning. As a relevant example, in commonsenseQA problem(Talmor et al., 2018), KagNet(Lin et al., 2019) views the concepts in the questions and answers as objects and effectively utilize external knowledge graphs to model their relations from both semantic and symbolic spaces. Impressive results are achieved when automatically extracting evidence from heterogeneous knowledge sources, and answering questions based on the extracted evidence(Lv et al.).

Adversarial Network Generative Adversarial Networks, i.e.GAN is a great success for generating pictures in computer vision field(Rajani et al.). Via an adversarial process, GAN treats the training procedure as a minimax game between a generative model and discriminative model. However, GAN works poor in natural language because it is not designed for discrete data. To solve this problem, SeqGAN(Yu et al.) models the data generator as a stochastic policy in reinforcement learning (RL). It bypasses the generator differentiation problem by directly performing gradient policy update, and makes strong improvements.

Pre-trained Model There is a long history of pre-training model for language representations. We briefly reviewed a few approaches related to our project in this section. Based on novel structure and powerful computation ability, the pre-trained `BERT` model(Devlin et al.) can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks. After exceeded by some models such as `XLNet`(Lv et al.), authors found that `BERT` was significantly undertrained and proposed a robustly optimized BERT pre-training approach, i.e. `RoBERTa`(Liu et al., 2019). In order to reduce the burden on GPU memory as well as speed up the training process, `ALBERT`(Lan et al., 2019) is proposed. It consistently helps downstream tasks with multi-sentence inputs. As a result, `ALBERT` establishes new state-of-the-art results on the GLUE, RACE, and SQuAD benchmarks while having fewer parameters compared to BERT-large.

3 Methods

3.1 Embedding Preparation for BERT

The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on original Transformer model(Vaswani et al., 2017).

Let $T = (s_1, ..., s_{|T|})$ denote the statements. For each statement, we have three options denoted by $c_i = (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$ and one label denoted by y_i. First, we process data to get the token embeddings and the segment embeddings.

For the token embeddings, we take $x_i^{(j)} = ([CLS], s_i, [SEP], c_i^{(j)}, [SEP]), (j = 1, 2, 3)$ as an input token sentence after adding special tokens and the WordPiece embedding of $x_i^{(j)}$ will be the token embeddings. We set the max-length as the max token embeddings' length. If the length of token embeddings is less than the max-length, we will add zeros to pad the length.

For the segment embeddings, if the position of the token embeddings belongs to $([CLS], s_i, [SEP])$, we set it to one; otherwise, we set it to zero.

Then, we use (token embeddings, segment embeddings, labels, ids) as the input of our model.

In our task, we get a worse result on BERT-large and BERT-large has slower speed than base, thus we discard the BERT-large model.

Finally, we can get the result after fine-tuning and print the wrong samples for analysis.

The specific structure will have small changes as methods change. And the input of the model will be correspondingly adjusted.

3.2 Leverage Extra Information

In this part, we leverage extra information, triples extracted by OpenIE tools and explanation auto-generated mentioned in Section 3.3.

3.2.1 Textual Joining on Extra Part

For baseline `BERT`, we concatenate each option after the statement, connected with a [SEP] label. In OpenIE triple experiments `BERT+OpenIE`, after extracting triples from the sentence and options, we combine the subject, relation and object together, forming simple sentences and add them to the original ones, separated by [SEP]. The max number of trp sentences considered is 3. In the third trial `BERT+Generation`, with our explanations generated from `UniLM`, we add it after the statement.

- Baseline input format: "[CLS] statement [SEP] option [SEP]".
- Triple input format: "[CLS] statement [SEP] state_trp_sentences [SEP] option [SEP] option_trp_sentences [SEP]".

- Explanation input format: “[CLS] statement [SEP] explain [SEP] option [SEP]”.

3.2.2 Featured Concatenation on Extra Part

In this section, our input consists of two parts, triple(or explanation) part and statement+option part.

For baseline `BERT+OpenIE`, we use the triples extracted from each option by OpenIE tools. We treat the statement and the option in the same way described in section above. They are put into `BERT` together. In the second trial `BERT+ Generation+ OpenIE`, we concatenate the explanation after the statement and the triple part remains the same with baseline. In the third trial `BERT+Generation`, we use our explanations generated from `UniLM` instead of triples. The statement+option part remains the same with baseline.

- Baseline input format: “[CLS] statement [SEP] option [SEP]” & ”[CLS] subject [SEP] relation [SEP] object [SEP]”
- Concatenating explanation input format: “[CLS] statement [SEP] explain [SEP] option [SEP]” & ”[CLS] subject [SEP] relation [SEP] object [SEP]”
- Featured explanation input format: “[CLS] statement [SEP] option [SEP]” & ”[CLS] explain [SEP]”

3.3 Explanation Generation

3.3.1 Basic Trial with UniLM

Up to now, UniLM(Dong et al., 2019) is still the state of the art model in pre-trained model for pre-trained language model-based generation architecture. Thus, we would like to use the pre-trained UniLM to generate the explanation based on the references in Commonsense Generation of SemEval 2020 Task 4-C, and use the explanations to improve the classification accurracy.

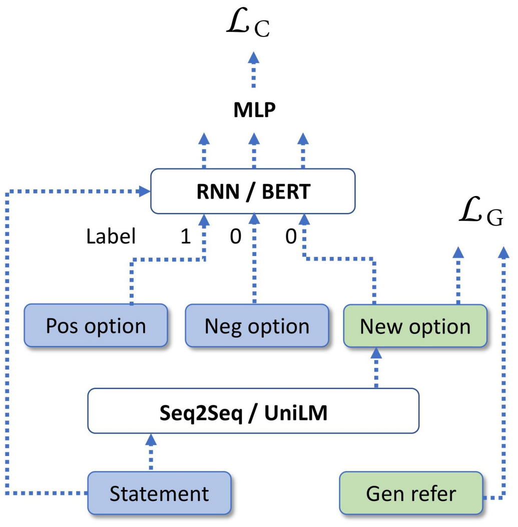

Figure 1: GAN Architectures

Model	Bleu1	Bleu2	Bleu3	Bleu4
Transformer	20.9	4.0	0.9	0.0
Seq2Seq	23.8	5.9	0.0	0.0
S2S+GAN	**33.8**	**8.4**	**1.4**	**0.1**
UniLM	44.8	23.8	11.2	5.0

Table 2: Generation Result

3.3.2 Further Exploration with GAN

We find the Commonsense Explanation task and Generation task are very suitable for the architecture of adversarial network. We change the general used 2-choices discriminator into multiple-choices discriminator to make the trained discriminator fit the Commonsense Explanation better. Thus, we can show the influence of adversarial training on generator and discriminator. We provide cross-entropy loss and negative discriminated loss for the generator. The optimizing goal for both models can be seen from the following formula.

$$\min_{G}\max_{D} L(D,G) = E_{x\sim p_r(x)}[\log D(x)] + E_{x\sim p_s(x)}[\log(1 - D(x) + G(x)]$$

The adversarial architecture can be seen from Figure 1. After the sentence is generated, we switch one of the wrong sentences in Explanation task and label it as a wrong sample. We have finished the naive version with Seq2Seq and RNN-classifier. From the result Table 2, we can find that the adversarial training does improve the performance of Seq2Seq and Transformer generator.

4 Experiments

4.1 Datasets

The dataset provided along with the project consists of 10,768 nonsense natural language statements in total that require human commonsense reasoning ability to choose the right explanation from wrong ones, where each statement has three optional explanations. We use the official split after duplicate removal, including 6989 for training, 1759 for development and 2020 for test.

As we choose to enhance the model with the help of natural language generation in SemEval 2020 Task 4-C, the corresponding dataset is also used in our project. The two dataset are well matched.

Since the dataset is different from the one in experiments in the origin paper(Wang et al., 2019), applying the same model, i.e. `BERT` results in a much higher accuracy, and we try to improve the performance based on this more challenging benchmark.

4.2 General Parameters Setting

We set the training batch size to 5 and evaluation batch size to 8. We use Adam (Kingma and Ba) with epsilon of 1e-8 for optimization. 2 training epochs is applied, and 4 GPUs are used in our experiments. For Seq2Seq and RNN-Classifier, we use 1 layer RNN with hidden dimension of 300 for both side. The training batch size is 64. The optimizer is SGD with learning rate of 1e-5 and weight decay of 1e-5. The word embedding is initialized with Glove (Pennington et al., 2014). We selected the best model in dev set within 100 epochs. For UniLM, We use Adam (Kingma and Ba) with learning rate of 2e-5 for optimization. The optimizing use 10 epochs.

5 Performance Analysis

All experiments are using the parameters mentioned in sections above, and are tested on development sets to pick the best-performing one. In `BERT` experiments, the best performance is reached when learning rate is 8e-5, 87.09% on development dataset and 78.71% on blind test dataset. Detailed information can been seen in Table 3.

In Textual Joining experiments in Table 5, the best performance is reached in `BERT+Geneation`, when learning rate is 4e-5, 87.66% on development dataset and 79.41% on blind test dataset. We successfully improve the accuracy compared with the baseline. Details can be seen in Table 4.

Adding explanation is more effective because many of the generated sentences actually are the reasonable explanation after the training of `UniLM`. For example, the generated explanation of the statement ”He dipped his chips in sand.” is ”Sand is not a liquid to dip a chip in.” The failure of adding triples to improve the performance might result from its simple extracting way of semantic relationship. And some of the empty triples also result in inconsistency of our input representation.

In Featured Concatenation experiments in Table 6, we set all the learning at 2e-5. In `BERT+OpenIE` experiments, the performance become worse. However, improve-

Learning-rate	Dev-Loss	Dev-Acc	Test-Loss	Test-Acc
2e-5	0.3656	86.81%	0.5557	77.82%
4e-5	0.3381	86.87%	**0.5277**	78.37%
6e-5	0.3378	87.38%	0.5372	77.97%
8e-5	0.3415	87.09%	0.5447	**78.71**%

Table 3: BERT Results

Learning-rate	Dev-Loss	Dev-Acc	Test-Loss	Test-Acc
2e-5	0.3672	86.75%	0.5584	77.97%
4e-5	0.3379	87.66%	**0.5282**	**79.41**%
6e-5	0.3478	87.89%	0.5891	77.77%
8e-5	0.3593	86.19%	0.5411	78.12%

Table 4: BERT+Generation Results

Model	Learning-rate	Dev-Loss	Dev-Acc	Test-Loss	Test-Acc
BERT (baseline)	8e-5	0.3415	87.09%	**0.5447**	**78.71**%
BERT+OpenIE	4e-5	0.3365	88.00%	0.5359	78.56%
BERT+Generation	4e-5	0.3379	87.66%	**0.5282**	**79.41**%

Table 5: Textual Joining Results

Model	Dev-Loss	Dev-Acc	Test-Loss	Test-Acc
BERT+OpenIE	0.3644	87.66%	**0.6145**	**78.47**%
BERT+Generation	0.3477	88.12%	**0.5934**	**79.11**%
BERT+Generation+OpenIE	0.3607	87.95%	0.6332	77.62%

Table 6: Featured Concatenation Results

ment is achieved when we extract the features directly from the generated explanation from `UniLM`, the performance is better than extracting features from triples and also better than the original `BERT` baseline.

6 Conclusion and Future work

We have enhanced the baseline BERT performance by generating explanation with UniLM. We also find the adversarial training could be very helpful for the commonsense generation with our RNN-based experiment. With those inspiring results, we can foresee that after being enhanced by adversarial training, UniLM would generate even better explanations and it would in turn help us to achieve an interpretable commonsense explanation with greater performance. We would leave it as our future work. We would like to extract more explicit representation of semantic relationship using semantic labeling (Zhang et al., 2019) in the future.

7 Acknowledgements

A portion of codes are provided by the senior in FUDAN NLP group (parts of BERT and OpenIE). Credit to Geng Lin for his code extracting triples from OpenIE.

References

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning.

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

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2019. Semantics-aware bert for language understanding.