Hook

Let's imagine this case: you are employed in an AI company, your boss comes and says:' Hey guys, I want you to develop a Q&A bot.' After several days, he comes again and says:' Team N, you are required to add a new function to the bot, sentiment analysis.' In this case, what will you do? One naive solution is to train a new model, and then add it to the bot. But there are some foreseen problems to your solution: the time and space of your model will be costly. Besides, your budget will increase a lot. This paper will give us a possible solution to this problem.

1. Prerequisites

1.1 Prompting(2'14'')

Before we understand this paper we need to know some basic concepts. Let's introduce 'prompt' first. So what is a prompt? It is like a guide in reality. But what it guides is our model. It helps the model to understand what it should do to address the task.
You may ask:' Why do we need a prompt for our task?' In practice we usually want our model to be powerful. It means that our model needs to have the ability to address different tasks, like NER tasks, Q&A bot, text generation tasks, etc.
And how does a prompt work in a model? What the prompt does in text is to generate an auxiliary sentence. Let's see this picture, we have a feature of the movie, then the prompt adds an introduction sentence to this sentence. In this case, our model relies on this prompt and understands it needs to generate a comment of this movie.

1.2 Autoencoding VS Autoregressive (3'15'')

Let's next introduce the Vanilla Transformer. The transformer has two parts: the encoder on the left and the decoder on the right.

1.2.1 Encoder handles the previous token and the following token at the same time. Each token can grasp information from the start to the end and vice versa. This makes it awesome for Natural Language Understanding tasks.

1.2.2 Decoder

Opposite to the encoder, the decoder is a unidirectional or autoregressive model. As it is trained on Masked Language Modeling, it naturally processes sequences either from left to right or right to left, depending on the implementation. So it is good at generating text.

1.3 Few-shot learning(4’30”)

Few-shot learning in natural language processing (NLP) refers to the ability of a model to understand and perform new tasks after seeing only a few examples. This is in contrast to traditional machine learning that often requires large datasets to achieve good performance.

Here is an example of 3 way 2 shot few shot.

The common way to utilize transfer learning and implement the few-shot learning in NLP is to fine-tune a pretrained model in a small dataset.

1.4 Zero-shot learning

Then I will give a brief introduction about zero-shot learning.

Zero-shot learning refers to the model's ability to understand and perform tasks that has not been explicitly trained to do. Let’s discuss the following example. The model doesn’t see the zebra before, zero-shot learning makes the model recognize the zebra by telling its shape, stripe and color based on previous experience.

2 Paper details (6’15”)

2.1 Overview

Let's discuss the central concept of this paper. The primary idea is to use a Pretrained Language Model to generate data without any fine-tuning, and then to fine-tune another PLM with this generated data to perform specific tasks.

In this paper, we will focus on the Natural Language Inference (NLI) task. This task requires a system to determine whether one sentence (the "premise") entails, contradicts, or is neutral with respect to another sentence (the "hypothesis").

Let's consider the following example: The initial sample sequence is "The opening date of the station was estimated to be mid-2020." Given the label y (entailment), we use the prompt "in other words". After inputting the sample sequence and prompt into the PLM, the output is "The station was to open in 2020." We then use the sample sequence, the generated text, and the label y to fine-tune another PLM."

2.2 Generation Data with Unidirectional PLM

There will be an issue that is generating repetitive words in the generating process. There are two ways to fix this problem, one is to modify the temperature feature in the softmax function. Another way is to get the likelihood as the score and get rid of lower-score items.

2.3 Fine-Tune bidirectional PLM

In the fine-tuning process, the training data might contain noise, although we have selected the high-quality data. This paper just slightly modifies the cross-entropy loss function, uses TD learning to implement KL divergence as the penalty or regularity.

3 experiments (8’20’’)

3.1 How to Run

Let’s discuss the experiments part. I have fixed the issue of the origin open source code and written detailed documents on this webpage. Just follow the steps if you want to run it.

3.2 Obstacles

I have faced lots of problems when I tried to run the project. Let’s go through the details.

3.2.1 Running Obstacle —— Outdated package

First one is the outdated package issue. This project includes lots of unnecessary packages, even some of them are wrongly imported. Just delete them. Then get help from your best friend — google.

3.2.2 Running Obstacle —— wrong function signature

And the second obstacle is the wrong function signature in the open source code. I don’t know why the author could run it with the wrong parameter set. It’s really weird.

What I did is to read the code and implement what the original code missed.

3.2.3 Running Obstacle —— RAM filling up

The last issue is extremely complicated. I will show it briefly.

I always get the string ^C on the output when I try to run the project by executing the command on the jupyter notebook, which means that executing the commands doesn't work. After checking the monitor, I found the RAM is filling up. But I ran the project on my machine on the terminal successfully.

Finally I found the reason: using ! to execute the command on the jupyter notebook make each command run in a new sub-process, separated from the python kernel running the notebook. The garbage collection must be much less efficient.

That's all for my part. Let's welcome Yihan.

4. Pros and Cons (9'40'')

4.1 Pros

So why is the paper important? Let's check out its good points:
First, both models are just the right size to fit into the research hardware.

Then, SuperGen removes the need for many cross-task annotations and gives the classifier PLM more training data than few-shot scenarios.

Also, zero-shot methods skip any training data from the test domain and perform inference on the target task directly.

Lastly, prompts guide a unidirectional PLM to make training data automatically, so no fine-tuning is needed.

4.2 Cons

But there are also 2 cons:
We may not have task-specific samples for tuning hyperparameters, and the quality of our generated training data may not be high enough, especially when a task is challenging, or the data distribution differs a lot from the pretraining data.

But don't worry – we've got a plan to address them: which is to extend SuperGen for few-shot scenarios and use some labeled data for better quality data and hyperparameter tuning.