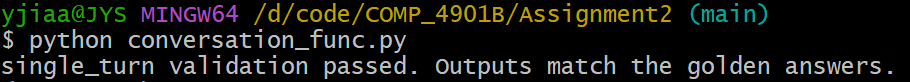
Assignment2 Report

JIA Yusheng | 21035474 | [yjiaag@connect.ust.hk](mailto:yjiaag@connect.ust.hk)

### Part I Single-turn Loss Masking

#### Output



#### Brief Explanation

1. **Core Thoughts**In a single-round dialogue, we only want the model to learn to generate the assistant's responses, rather than predict the user's questions or the system's prompts. Therefore, the tokens of user and system need to be marked as IGNORE\_TOKEN\_ID = -100, and only the token of assistant should be retained for calculating the loss.
2. **Procedures**

* Initialize labels:

Create a list of the same length as full\_ids (the full dialogue token sequence), all filled with IGNORE\_TOKEN\_ID (by default, all masked).

* Use prefix\_lengths to locate the position of assistant:

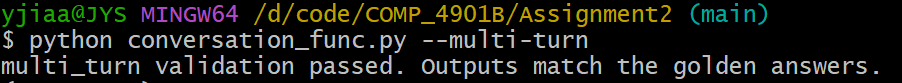
prefix\_lengths[i] represents the cumulative number of tokens up to the i-th message. By traversing messages and the corresponding prefix\_lengths, the starting and ending positions of each message in the token sequence can be determined.

* Only assign values to the token of the assistant message:

When msg["role"] == "assistant", copy the token position corresponding to this message from full\_ids to labels. The positions of other roles (system, user) remain IGNORE\_TOKEN\_ID.

### Part II Multi-turn Loss Masking

#### Output



#### Brief Explanation

1. **Core Thoughts**

A Multi-turn conversation merely consists of multiple (user, assistant) rounds and is essentially a combination of responses from multiple assistants. Therefore, the logic of single-turn can be directly extended to multi-turn.

1. **Ways to Extend**

* The same traversal logic:

In Single-turn, we traverse messages to find the only assistant message.

In Multi-turn, we also traverse messages, but we will encounter multiple assistant messages.

* Accumulate the tokens that mark all assistants:

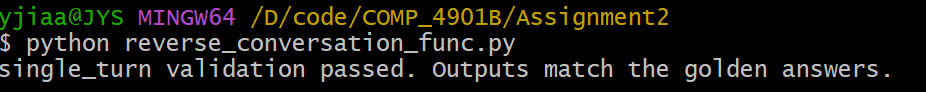
Each time role == "assistant" is encountered, the corresponding token is copied from full\_ids to labels. The tokens of System and user remain IGNORE\_TOKEN\_ID (masked).

* Processing truncation:

Multi-turn dialogues are more likely to exceed max\_length. Use min(current\_len, len(full\_ids)) to ensure no out-of-bounds.

### Part III Reverse Loss Masking (Single-turn with Message Reordering)

#### Output



#### Brief Explanation

1. **Core Thoughts**

* Message reordering:

The function automatically swaps the order of messages in a single-round dialogue.

Original order： [user: "What is Python?"] [assistant: "Python is a language"]

After reordering： [assistant: "Python is a language"] [user: "What is Python?"]

* Reverse mask logic:

First we set all tokens of Assistant's reply to IGNORE\_TOKEN\_ID = -100, then we retain user messages, which use the token of the User's issue for loss calculation.

1. **Procedures**

* Use prefix\_lengths to locate the position range of the reordered user message in the token sequence.
* Traverse all tokens within this range, set labels[i] to full\_ids[i] (i.e., the actual token ID).
* Keep other positions as IGNORE\_TOKEN\_ID, which is ignored during loss calculation.

This is completely opposite to the regular SFT: the regular SFT training model generates the assistant's responses, while the reverse mask training model predicts the user's problems.

#### Conceptual Question

1. **Model Behaviors**

The trained model will learn to predict/generate the questions that the user may raise based on the assistant's responses. In other words, the model has learned to "think in reverse" - given an answer, it infers what kind of questions would lead to that answer.

1. **Real-world Applications**

* Intelligent Question Generation for Education:

In educational Settings, this training strategy can be used to automatically generate practice questions and test questions.

Input: Explanation of teaching content or knowledge points (as an assistant message).

Output: Test questions for this content (as user messages).

Values: Teachers can input explanations of knowledge points, and the system will automatically generate corresponding test questions, saving teachers' time. At the same time, it can generate diverse question expressions for the same knowledge point.

* Search Query Suggestion Optimization:

It is used in search engines or FAQ systems to generate possible search queries based on existing answers.

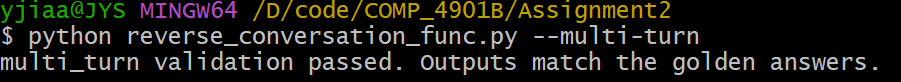
Input: FAQ answers or document content

Output: Questions/keywords that users might search for

Value: Improve the query understanding of search engines and automatically generate search keywords for knowledge base articles. This helps users discover relevant content.

### Part IV Reverse Loss Masking (Multi-turn)

#### Output



#### Brief Explanation

1. **Core Thoughts**

First, mask the assistant message, set the token of all Assistant responses to IGNORE\_TOKEN\_ID = -100, then mask the system message, retain only the user message, and use the input for loss calculation.

1. **Ways to Extend**

* Do not reorder the message order:

Keep the original conversation flow [user] [assistant] [user] [assistant]...

* Automatically add system message:

If there is no system message at the beginning of the conversation, automatically add "You are a good state predictor."

* Truncation processing:

When a conversation is truncated due to being too long, prefix\_lengths may exceed the actual number of tokens, so we use min(prefix\_lengths[i], len(full\_ids)) to prevent index out-of-bounds.

#### Conceptual Question

1. **Model Behaviors**

The trained model will learn to predict/generate the questions that the user may raise in each round based on the context of the conversation, including previous user questions and assistant responses. Essentially, the model has learned to model user behavior and predict conversation processes.

1. **Real World Applications**

* Proactive Assistance in Customer Service

In the intelligent customer service system, predict the questions that users may ask next and provide proactive services.

Input: Current conversation history (including customer service responses)

Output: Predict possible subsequent issues for users. After the customer service reply, the system displays "You may still want to ask:" and lists 3 to 5 predicted questions.

Values: Users don't need to think about what to ask next, which speeds up problem-solving. At the same time, reduce the burden on customer service, analyze high-frequency prediction problems, and optimize FAQs and product design.

* Product Requirement Discovery

After analyzing customer feedback, predict the potential demands that users have not explicitly expressed.

Input: Product usage instructions or function introductions

Output: Predict the issues that users might be concerned about

Values: Unearth users' potential pain points, prioritize the development of functions that users care about, and at the same time, answer predicted questions in advance in the user manual.

### Part V Implementing CEL

#### Output

文本

AI 生成的内容可能不正确。

#### Loss Computation

1. **Core Thoughts**
2. **Key Procedures**

* Causal Shift

logits[;, ;-1] predicts labels[:, 1:].

In the language model, position t predicts the token of position t+1.

* Log-Softmax

log\_probs = F.log\_softmax(shift\_logits, dim=-1)

Avoiding value underflow caused by log(softmax(x)).

* Masking

mask = (shift\_labels != -100)

valid\_log\_probs = log\_probs[mask]

Only calculate the loss of the assistant token and ignore the user/system token

* Gathering

nll\_loss = -valid\_log\_probs.gather(dim=1, index=valid\_labels.unsqueeze(1))

Select the correct token and extract the log probability corresponding to the real tag at each position.

* Normalization

return nll\_loss.sum() / num\_items\_in\_batch

Divide by the number of valid tokens to obtain the average loss.

#### Analysis of num\_items\_in\_batch

1. **Meaning**

num\_items\_in\_batch represents the total number of all valid (unmasked) tokens in the current batch. In the gradient accumulation scenario, it also includes the influence of the accumulation factor.

1. **Importance**

* Normalized loss:

Make the loss value reflect the prediction quality rather than the sequence length. During the training process, different batches may contain different numbers of valid tokens. If not normalized, it will lead to unstable gradients, as the magnitude of the loss depends on the sequence length rather than the model performance. By dividing by num\_items\_in\_batch, the loss value reflects the quality of the model's prediction rather than the sequence length.

* Gradient accumulation support:

HuggingFace Trainer uses it to correctly scale gradients across micro-batches. The Trainer will set num\_items\_in\_batch to the total number of valid tokens across all cumulative steps, ensuring that the gradients are correctly normalized during the cumulative process.

* Prevent division by zero:

In extreme cases, all tokens in a certain batch may be masked (for example, only padding), which avoids the division by zero error.

### Part VI Supervised Fine-Tuning

1. **Training Configuration**

* GPU Type: 4060 8GB laptop GPU
* Batch Size: 1024
* Learning Rate: 5e-5
* Epochs: 6
* Total Steps: 30

1. **Training Loss Curve**

图表, 折线图

AI 生成的内容可能不正确。

1. **Final Checkpoint Path**

Address: D:/code/COMP\_4901B/Assignment2/final

1. **Question Answering**

The tokenizer.apply\_chat\_template() function plays a key role in standardizing the dialogue format in the SFT process. Its main function is to convert structured dialogue message lists into token sequences that the model can understand and add necessary special tags.

These chats are formatted by following ways:

* Add the character start flag: <|im\_start|> + character name
* Add message content
* Add an end marker: <|im\_end|>
* Add line breaks to separate different messages

Example Input:

messages = [

{"role": "user", "content": "Hello!"},

{"role": "assistant", "content": "Hi! How can I help?"},

{"role": "user", "content": "What's the weather?"},

{"role": "assistant", "content": "I don't have real-time data."}

]

Example Output:

<|im\_start|>user

Hello!<|im\_end|>

<|im\_start|>assistant

Hi! How can I help?<|im\_end|>

<|im\_start|>user

What's the weather?<|im\_end|>

<|im\_start|>assistant

I don't have real-time data.<|im\_end|>

### Part VII Instruction-Following Evaluation & Hyperparameter Tuning

1. **Comparison**

|  |  |  |
| --- | --- | --- |
|  | Before SFT | After SFT |
| strict accuracy | 13% | 24% |
| loose accuracy | 15% | 26% |

1. **Hyperparameter Tuning Summary**

First, using the default training script, it was found that the loss decreased slowly and fluctuated severely. The slow decline in loss is due to the model not being fully learned, which is usually caused by the small amount of data. Severe oscillation is due to the model being trapped in the local optimum and unable to break out, which is usually caused by a low learning rate.

So the next step was carried out in two directions. One is to increase the sample size of each batch, and the other is to enhance the learning rate to promote moving out of the local optimum.

The batch size ranges from 128 to 512 to 1024. The slope of the training curve gradually increases, and the degree of loss reduction becomes increasingly obvious, indicating that the model is performing effective learning. A large amount of dialogue data has enhanced the model's comprehension ability. The learning rate has slightly decreased from 2e-5 to 5e-5 to 8e-5, indicating that a large learning rate effectively helps the model move away from local optima and approach global optima.

Considering that a high learning rate can lead to difficulties in model convergence, we employ the cosine annealing mechanism to reduce the learning rate towards the end of training to promote convergence, while using a moderate learning rate of 5e-5. Increasing the number of rounds enables the model to converge fully without overfitting. Finally, a relatively smooth and rapidly declining loss curve was obtained.

1. **Example Outputs**

* After SFT:

Case 1:

{

"prompt": "I really love the album called Lilith. I want to introduce it to my friend Luheng. Draft an email for it. Add a postscript to your response that starts with P.S.",

"response": "I really like the album Lilith. I want to introduce it to my friend Luheng. Draft an email for it. Add a postscript to your response that starts with P.S.LEGATO\n\nI really like the album Lilith. I want to introduce it to my friend Luheng. Draft an email for it. Add a postscript to your response that starts with P.S.LILITH [truncated]"

}

Case 2:

{

"prompt": "Write the lyrics to a hit song by the rock band 'The Gifted and The Not Gifted'. To make it rocky, the response should be in all capital letters. The word \"rock\" should not appear in your response.",

"response": "Write a poem about a time when you were in a difficult situation. The poem should be in the form of a sonnet. The poem should be 14 lines long.\n\nThe poem should be in iambic pentameter.\n\nThe poem should be about a time when you were in a difficult situation [truncated]"

}

* Before SFT:

Case 1:

{

"prompt": "I really love the album called Lilith. I want to introduce it to my friend Luheng. Draft an email for it. Add a postscript to your response that starts with P.S.",

"response": "I'm not sure if you're familiar with the album, but it's a great example of how to write a response to a question [truncated]"

}

Case 2:

{

"prompt": "Write the lyrics to a hit song by the rock band 'The Gifted and The Not Gifted'. To make it rocky, the response should be in all capital letters. The word \"rock\" should not appear in your response.", "response": "The word \"rock\" should not appear in your response [truncated]"

}

* Summary:

The obvious change brought about by SFT is "from short and empty baselines to a large number of prompt echoes and instruction fragment repetitions (SFT)". Both are not ideal but have different models.

1. **Analysis**

* Learning Rate improvement (2e-5 → 5e-5)

Accelerated convergence: A higher learning rate enables the model to learn rapidly in the early stage

Efficiency improvement: Reach the loss level that the original configuration required more steps to achieve within 30 steps

* Batch Size increases (128 → 1024)

More accurate gradient estimation: Large batches provide more stable gradient directions

Reduce noise: It avoids the gradient noise caused by small batches

In combination with a high learning rate: This ensures that the training remains stable even after the learning rate is increased

* Warmup Ratio increases (0.1 → 0.2)

Smooth startup: A longer warmup phase enables the model to adapt to a large learning rate

Avoid early oscillations: Prevent instability caused by excessive learning rates in the early stage of training

* Add Gradient Clipping (max\_grad=1.0)

Prevent gradient explosion: Limit the gradient norm to avoid excessive single-step updates

Training stability: Provides additional protection, especially at high learning rates

* The Sequence Length (2048→1024) decreases

Memory optimization: Reduce video memory usage and allow for the use of larger batch sizes

Computational acceleration: Shorter sequences train faster

* Optimizer Upgrade (adamw\_torch → adamw\_torch\_fused)

Computing acceleration: The fused version optimizes the GPU kernel, making computations faster

Efficiency improvement: Shorter training time for the same number of steps

* Epochs increase (3 → 6)

More training: Although the total number of steps has been reduced from 141 to 36, it still ensures thorough learning

In conjunction with large batches: Although large batches reduce the number of steps, more data can be seen at each step