Guide to Running and Evaluating the IFEval Benchmark

1 Introduction

This guide explains how to use the provided `nlp_ifeval.ipynb` notebook to generate responses for the IFEval benchmark and evaluate the generated responses. Students will customize the notebook to load their fine-tuned model and optimize parameters such as batch size and maximum response length. The provided files and scripts are designed to work efficiently in Google Colab within a runtime of approximately 1 hour and 40 minutes. However, students can stop the generation process earlier to evaluate a subset of prompts.

2 Files Provided

The following files are included in the provided zip file:

- `nlp_ifeval.ipynb`: Notebook to generate IFEval responses.
- `ifeval_command.txt`: file with the command from section 4.1 in order to copy paste it.
- `instruction_following_eval/`: Folder containing scripts for evaluating the responses.
  - `data/`: Folder with input prompts and placeholder for results.
  - `evaluation_main.py`: Script to evaluate generated responses.

3 Instructions for Generating Responses

Open the `nlp_ifeval.ipynb` notebook in Google Colab or a local Jupyter Notebook environment. Modify the following section to load your fine-tuned model:

```
model = AutoModelForCausalLM.from_pretrained("your-finetuned-model")
```

Adjust the `max_length` and `batch_size` parameters to balance between performance and resource usage:

- `max_length`: Controls the maximum length of the generated responses. Larger values increase time and memory usage.
- `batch_size`: Defines the number of prompts processed at once. Larger values require more VRAM.

Take into account that the give notebook was evaluated in Colab using a 7B model with nf4 quantization and fp16 computational dtype in 1 hour and 40 minutes. You might want to stop the generating process in a given batch and evaluate a subset of the prompts. The IFEval benchmark that we have provided allows you to do that, but do not try it with the one from google-research github.

The generated responses will be saved in `instruction_following_eval/data/input_response_data.jsonl`.

4 Evaluating the Responses

4.1 Step 1: Open a WSL Terminal

Navigate to the parent directory of the `instruction_following_eval` folder using a WSL terminal.

4.2 Step 2: Execute the Evaluation Script

Run the following command to evaluate the generated responses:

```
python -m instruction_following_eval.evaluation_main \
    --input_data instruction_following_eval/data/input_data.jsonl \
    --input_response_data instruction_following_eval/data/input_response_data.js
    --output_dir instruction_following_eval/data/evaluation_results
```

The script will compute evaluation metrics and save them in the `evaluation_results` folder. You have this whole command in the file `ifeval_command.txt`.

5 Metrics Description

After running the evaluation, two key metrics are provided in `eval_results_loose.jsonl`:

- **Prompt-level Accuracy**: Measures how accurately the model's responses match the expected output at the prompt level.
- **Instruction-level Accuracy**: Measures accuracy at the instruction level, accounting for all related prompts.

Additional detailed metrics include:

- **change_case**: Evaluates the model's ability to handle capitalization and case transformations.
- **combination**: Assesses how well the model combines information from multiple prompts.
- **detectable_content**: Verifies the presence of placeholders and specific content.
- **detectable_format**: Checks response formatting (e.g., JSON format, bullet lists).
- **keywords**: Evaluates the inclusion and frequency of required or forbidden words.
- **language**: Measures the language correctness of the response.
- **length_constraints**: Assesses adherence to constraints on the number of paragraphs, sentences, and words.
- **punctuation**: Evaluates proper punctuation use.
- **startend**: Verifies if the response starts or ends as required.

The same metrics are obtained from strict evaluation.

5.1 Example Evaluation Results

An example of the evaluation output is shown below:

```
Instruction-level Accuracy: 0.8557
Prompt-level Accuracy: 0.7963
change_case: 0.8427
combination: 0.7692
```

```
keywords: 0.8650
language: 0.9677
...
```

Only the **prompt-level** and **instruction-level** metrics from the loose evaluation are used for grading purposes. The metrics in the example are obtained from GPT-4 and you should not aim to reach those.

6 Strict vs Loose Evaluation

When evaluating the responses generated by your fine-tuned model, the IFEval benchmark provides two types of evaluation metrics: **strict evaluation** and **loose evaluation**. These two approaches differ in how they assess the accuracy and correctness of the model's outputs.

6.1 Strict Evaluation

Strict evaluation directly checks if the model's response adheres to the verifiable instructions without any transformations or modifications. The response is marked as `True` if it exactly matches the required format and satisfies all conditions specified in the instruction. Any deviation, no matter how minor, will result in the response being marked as `False`.

Example:

- **Instruction:** "End your email with: P.S. I do like the cake."
- **Response 1 (Correct):** `"P.S. I do like the cake."`
- **Response 2 (Incorrect):** `"P.S. **I do like the cake**"` (Markdown formatting makes it incorrect.)

Strict evaluation is ideal for tasks requiring exact compliance with formatting and specific rules, such as generating structured data or programmatic responses. However, it is less forgiving of minor formatting issues that do not significantly affect the meaning or intent of the response.

6.2 Loose Evaluation

Loose evaluation is a more flexible approach that mitigates false negatives by applying a series of transformations to the response before checking compliance with the instructions. A response is marked as `True` if any of the transformed versions match the required conditions.

Transformations Applied:

1. Removing markdown tags (e.g., `*`, `**`).
2. Skipping the first line of the response (e.g., `"Sure, here it is:"`).
3. Skipping the last line of the response (e.g., `"Hope it helps."`).
4. Combining the above transformations to create eight possible variations of the response, including the original.

Example:

- **Instruction:** "End your email with: P.S. I do like the cake."
- **Response:** `"P.S. **I do like the cake**"`
- **Loose Evaluation Result:** `True` (Markdown tags are removed during transformation, making the response valid.)

Loose evaluation is better suited for natural language tasks where semantic correctness is more important than strict adherence to formatting rules.

6.3 Which Evaluation is Used?

For the IFEval benchmark, the **loose evaluation metrics (prompt-level and instruction-level accuracy)** are the primary metrics considered for grading purposes. These metrics better reflect the practical utility of the model's outputs in real-world applications, where minor formatting errors may not significantly impact the response's quality or usability. However, students are encouraged to review the strict evaluation metrics as well, as they highlight the model's ability to follow precise instructions without deviation.

7 Recommendations

- Test with smaller `batch_size` and `max_length` values to ensure compatibility with the available hardware.
- Evaluate a subset of prompts if time or resources are limited.
- Document any changes made to the notebook for reproducibility.