**Code Generation Results - HumanEval  
SLMs without fine-tuning**   
December 7, 2024

1. **Challenges**

* The Replicate API library would not work directly, so I had to use its version within another library – LangChain.
* I used the code from this repo for HumanEval evaluation of all models: <https://github.com/openai/human-eval>. The multiprocessing module executed with an error: "AttributeError: Can't pickle local object in Multiprocessing". **Had to modify the original code to fix it**.
* The final check of the code correctness in the original OpenAI code is done by combining the problem (otherwise called starter code or function docstring) and completion: **problem["prompt"] + completion** which means the completion shouldn’t contain the problem. Two issues: 1) incorrect indentation when joining the problem and completion, 2) some models may still misunderstand and include problem into completion (Llama 3 Instruct (trained for chat) does it for 33% of cases). Therefore, I am using an additional prompt to ask LLMs to **include the problem definition (function docstring) into the completion**, and I exclude **problem["prompt"]** from the string to be evaluated in the end. **Had to modify the original code**.
* **Summary of code modifications** (all in execution.py):
  + **Add class DillProcess** to fix the pickling issue (uses dill instead of pickle).
  + **Modify function check\_correctness()** to have an extra argument use\_prompt which controls the exclusion or addition of problem["prompt"] to the Python program to be run. The body of the function is modified to accommodate for the use of use\_prompt.
  + Modify exception handling to **add error tracebacks** (helps when the error message is empty).
* SLMs tend to output **additional explanations** and clarifications like: “Here is the requested code completion:” etc. which break the automatic code execution during the verification stage. Adding more specific instructions like: “Complete the following code. Output only the runnable code and nothing else:” would still lead to non-runnable content like triple backticks in the output. As a result – **I had to provide minimum help to some models** by removing the initial or trailing triple backticks or strings like “```python”. Or by adding “from typing import List” as this was removed in the process (when LLM forgets to include it into the repeated func definition)
* **General approach to evaluate all models**: create one extensive and comprehensive prompt for all models. If any model fails to fully understand it and outputs code with human phrases or non-runnable symbols, it should be considered the drawback of the model.

1. **Results**

* **Llama 3 8B** – promising results.
* Non-chat optimized model ”**meta/meta-llama-3-8b**” - several cases of hallucinations when functions are repeated and the code is incomplete in the end (stops at the middle of a function). See Appendix
* **Nous-hermes-2-solar-10.7b** – tries to explain the solution if no prompt is used (func docstring as prompt) – not runnable. 25.61% when using a prompt.
* **Gemma 7B** – incomprehensible output whether I include the prompt prefix or not.
* **Mistral** – version is required, but it is not provided on the Replicate website
* **Mixtral (MoE)** – not available on Replicate
* **Phixtral** – generates code, but contains extraneous text (Here is a solution). If I do additional post-processing, this will be a disadvantage for other models. Also, it is very slow – up to 2 minutes per test case (5 hours for the entire run)
* **GPT-J-6B** – not fit for the task as the model is too weak, outputs hallucinations that remind of the expected output only very remotely (trained in 2021).
* **Yi-6B** is a bilingual (Chinese) model – pass@1 = 3% if not using prompt (function docstring as prompt), otherwise if using a prompt the model outputs some irrelevant snippets of code and. Asking to output the starter code concatenated with the completion doesn’t help – the output still includes the completion without the beginning in most cases (<https://huggingface.co/01-ai/Yi-6B> ).
* **Flan-T5** outputs complete nonsense that resembles code – completely not runnable.
* **Phi** – not designed for code completion. Outputs incomprehensible combinations of letters (“em”, “emlen”, “A”, “A.A.A.A.”, etc.) as generated code with or without a prompt (if with prompt, the model repeats the entire prompt before the incomprehensible output).
* **Phixtral-2x2\_8** – MoE of two Phi models (4.5B), follows the instruction much better than Phi, reached Pass@1 = ~15%. Still outputs irrelevant human-like output although asked specifically not to do that: e.g. here’s the code, here’s the concatenated code, etc.
* **Mamba 2.8B**: if not using a prompt (func docstring as prompt) – the model tries to generate a completion, but then follows a paragraph of hallucinations that look like human free-form text with how-to questions about software development. When using a prompt – the model doesn’t even try to complete the code – it starts hallucinating right away (see saved file with examples).
* **Mistral 7B** provided the expected result. The model would strip any docstrings from the functions – only the definition def was left. I helped the model by removing triple backticks from start / end, “```python”, and adding “from typing import List” because the model would strip this import most of the times while the import is specific to the HumanEval dataset.
* **Mistral 3B**
* **SmolLM** -

All models received slight help by stripping ``` backticks at edges including the ```python string + adding “from typing import List” which is often stripped by SLMs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Hosted By** | **Model Size** | **Human-Eval**  **Pass@1** (Me / Leaderboard) | **MBPP** | **Comments** | **Cost ($)** |
| Llama 3 8B | replicate.com | 8B | 51.5% |  |  | 0.29 |
| Qwen | replicate.com |  | 43.9% |  | 200-300 s per one API call. | 3.55 |
| Nous-hermes-2-solar-10.7b | replicate.com | 10.7B | 25.61% |  |  | 0.61 |
| Phixtral-2x2\_8 (4.5B) | replicate.com | 4.5B | 14.64% |  | ~1 min per API call | 2.77 |
| Yi 6B | replicate.com | 6B | 3% |  | Function def + doc string as prompt | 0.44 |
| Mistral 7B | misral.ai | 7B | 31.1% / 30.5% |  |  | 0.01 |
| Ministral 3B | misral.ai | 3B | 64.63% / 77.4% (instruct) |  |  | 0.01 |
| Ministral 8B | misral.ai | 8B | 72.56% / 76.8% (instruct) |  |  | 0.01 |
| Codestral Mamba | misral.ai | 7.3B | 75.61% / 75% |  |  | 0.02 |
| Codestral latest | misral.ai | 22.2B | 26.83% / 81.1% |  |  | 0.15 |
| Mistral-Nemo-Instruct-2407 | misral.ai | 12B | 58.54%/ 67% |  |  | 0.01 |
| Mistral-Small-2409 | misral.ai | 22B | 70.73% / 80% |  |  | 0.03 |
| Mixtral-8x7B-v0.1 | misral.ai | 12 active (47 total) | 16.46% / 40.2% |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Gemma 7B | replicate.com |  | 0 % |  | Incoherent output | 0.05 |
| Gemma 2B | replicate.com |  | 0 % |  | Incoherent output | 0.05 |
| Flan-T5 | replicate.com |  | 0% |  | Incoherent output |  |
| Phi | replicate.com |  | 0% |  | Incoherent output |  |
| Mamba 2.8B | replicate.com |  | n/a |  | Incoherent output | 0.02 (20 calls) |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Conclusions**

Conclusions