Code Generation Results Summary

SLMs without fine-tuning

January 25, 2024

1. Code used to generate the below results

Code: https://github.com/agnedil/code-generation

2. 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". **I 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 – the code generates error when running, 2) some models may still misunderstand and include the problem into the 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 evaluated string. **I had to modify the original code to fix this**.
- **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.

3. 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.
- **Code Gemma 7b IT** (when asked to output the full func for HumanEval) – a) code generation template (per HG docs): unusable output – patchy pieces of code, sometimes 1 or 2 random lines, b) chat generation template: more usable output, but still a lot of errors in the first 5 problems: repeats def in the end, 2 out of 3 outputs were completions w/out func signature and docstring, 2 others contained extraneous text, e.g. the word "def" after the func was already provided, etc. Decided not to waste compute units – the leaderboard performance is still only 55%.
- **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. Also, it takes ~1 min per API call which is a lot, considering there are 500 data points in the MBPP dataset.
- **Qwen1.5-7b** (replicate.com) – demonstrated a good result on HumanEval Pass@1 at ~44%, but only 20% on MBPP. The main challenge with this model is that it takes 200-300 s per one API call - took 1 day to run MBPP on replicate. This is unacceptable for experiments with agents as I will have to make several API calls per one agent call + run this for all 500 MBPP data points again – will take more than a day per experiment.
- **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.
- **Codestral Mamba** – showed the best result on my leaderboard, followed by **Ministral 8B** and, surprisingly, **Ministral 3B**. The latter is the smallest model that I tried, but it outperformed many other models that are 2 to 2.5 times bigger, and even the models with 22B

parameters (7+ times bigger). Other models from the Mistral family also showed good results which, on average, make this group of models as the leading one among all other models.

- **OpenCodeInterpreter-DS-6.7B** and **Artigenz-Coder-DS-6.7B** – when asked to output the entire function, keeps saying "Here is the completed function" (even if I ask not to do it in the prompt). Model needs extra help by getting the code placed between ```*python* and ```. **May be better at pure code completion**?
- **Mamba 2.8B (**replicate.com): 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).
- Gemma 7B, Gemma 2B, Flan-T5, Phi, Mamba 2.8B (replicate.com) – incoherent output.
- **Deepseek-Coder-6.7B-Instruct** – scored great on HumanEval, but *did only 1% on MBPP*, mainly because the model outputs unnecessary explanations, although it is explicitly asked not to do that. Example: "Sure, here is the Python function that calculates." This is done for every data point. Somewhat similar numbers are for OpenCodeInterpreter-DS-6.7B. Reason is same: unnecessary clarifications when asked not to do it: "Here is the Python function that satisfies the given tests:" *Solution – maybe decrease temperature*?
- **Llama 3**.1 8B Instruct – released fall 2024. Inference takes an average of 2 minutes for Human Eval and 0.75 min for MBPP. Both tasks required 4 hours to finish running in Google Colab on an A100 GPU which is the best available. This may be too long for subsequent experiments, but I was able to get this model work for the Reflection workflow – **TODO: add timer for the entire notebook**.
- **Phixtral** on Replicate – takes 100 to 200 seconds per API call. Running this model for Big Code Bench (500 data points) took well all night and up to the lunch time of the next day. Considering that the results from this model are very low in general, I will discontinue using it for agent experiments as they will take even more time due to several API calls per iteration. Maybe use the HuggingFace version of the model? What if the HF version is more up-to-date and faster?

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.

The pass@1 scores below are provided for my run (first number) and then for the result shown on the Big Code Leaderboard (second number), if it is available:
https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard

Table 1. Pass@1 Score for Testing SLMs on Multiple Datasets

Model	**Hosted By**	**Model Size**	**Human-Eval Full Func** (Me / Big Code)	**H-E Compl**	**MBPP**	**LBPP**	**Big Code Bench**	**Rank**	**Temp / top_p**	**Cost $, full func**	**Notes**
Small Language Models (SLMs)											
Nxcode-CQ-7B-orpo	Google Colab	7.25B	82.93 / 87.23	75.61%	73%	22.84%	24%	1	1.0 / 1.0	$50/ month	

Model	**Hosted By**	**Model Size**	**Human-Eval Full Func** (Me / Big Code)	**H-E Compl**	**MBPP**	**LBPP**	**Big Code Bench**	**Rank**	**Temp / top_p**	**Cost $, full func**	**Notes**
Codestral Mamba	mistral.ai	7.3B	75.61% / 75%	60.37%	39.4%	26.54%	23%	3	0.7 / 1.0	0.02	
Ministral 8B	mistral.ai	8B	72.56% / 76.8% (instruct)	71.34%	56.2%	22.22%	24.6	2	0.3 / 1.0	0.01	
Deepseek-Coder-6.7B-Instruct	Google Colab		65.24% / 80.22%	70.73%	1%	0% (extra words!)	32.2%	9	1.0 / 1.0	$50/m	
Ministral 3B	mistral.ai	3B	64.63% / 77.4% (instruct)	61.59% / 77.4% (instruct)	51.8%	20.99%	26.8%	5	0.3 / 1.0	0.01	
Mistral-Nemo-Instruct-2407	mistral.ai	12B	58.54%/ 67%	53.05%	47.4%	21.6%	17.2%	7	0.3 / 1.0	0.01	
Llama 3.1 8B Instruct	Google Colab	8B	65.9% / 72.6%	55.47%	56.8%	21.6%	29.8%	4			
CodeQwen1.5-7B-Chat	Google Colab		50% / 87.2%	54.88%	55.2%	19.75%	26.8%	6	1.0 / 1.0	$50/m	
OpenCodeInterpreter-DS-6.7B	Google Colab		41% / 73.2%	71.95 / 73.2%	5.4%	8%	32.2%	8	1.0 / 1.0	$50/m	
Mistral 7B, open-mistral-7b	mistral.ai	7B	31.1% / 30.5%	35.98% / 30.5%	13.6%	9.9%	15.6%	11	0.7 / 1.0	0.01	
Nous-hermes-2-solar-10.7b	replicate.com	10.7B	25.61%	28.65%	30.4%	8%	43.4%	10	0.95 / 1	0.61	
Phixtral-2x2_8 (4.5B)	replicate.com	4.5B	14.64%	34.756%	14.6%	4.9	20.6%	14	0.95 / 1	2.77	
Artigenz-Coder-DS-6.7B	Google Colab		1.22% / 70.89%	73.17%	0.2%	4.32%	13.4%	13	1.0 / 1.0	$50/m.	

Model	Hosted By	Model Size	Human-Eval Full Func (Me / Big Code)	H-E Compl	MBPP	LBPP	Big Code Bench	Rank	Temp / top_p	Cost $, full func	Notes
Code Gemma 7b IT	Google Colab	7B	0% (? model)	27.44%	51% (chat model)	19.14% (chat model)	2.4%	12	1.0 / 1.0		
Slightly Bigger SLMs											
Mistral-Small-2409	mistral.ai	22B	70.73% / 80%	64.63%	60.2%	25.3%	20.6%		0.7 / 1.0	0.03	
Codestral latest	mistral.ai	22.2B	26.83% / 81.1%	64.63%	37%	48.15	12.8%		0.7 / 1.0	0.15	
Mixtral-8x7B-v0.1	mistral.ai	12B active (47B total)	16.46% / 40.2%	33.54%	0%	5.6%	11.8%		0.7 / 1.0	0.05	
Not Useful SLMs											
Qwen1.5-7b	replicate.com	7B	43.9%	200 s per API call	19.4%	200 s per API call	200 s API call	200 s API call	0.95 / 1	3.55	
Llama 3 8B	Replicate	8B	51.83% / 45.65%	API Error	API Error	API Error	API Error		0.95 / 1	0.29	
Yi 6B	replicate.com	6B	3%	3%	0.2%				0.95 / 1	0.44	
Gemma 7B	replicate.com	7B	0 %	0%	0%			0 %	0.95 / 1	0.05	
Gemma 2B	replicate.com	2B	0 %	4.26%	4.6%				0.95 / 1	0.05	
Flan-T5	replicate.com		0%						0.95 / 1		
Phi-2	replicate.com		0%	Incoherent (even if temp=0.25					0.95 / 1		
Mamba 2.8B	replicate.com	2.8B	n/a	Incoherent (even if temp=0.25	0%			0	0.95 / 1	0.02 (20 calls)	

Notes

1. The second number in the HumanEval Full Funk column is the performance of a model on the HuggingFace's Big Code Models Leaderboard (https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard) – not to be confused with the Big Code Benchmark dataset because HuggingFace presents only the humanEval results in its leaderboard.
2. HuggingFace transformer models' default temperature and top_p parameters are explained here: https://huggingface.co/docs/transformers/v4.22.2/en/main_classes/text_generation. Usually they are 1.0 and 1.0, respectively, and can be checked by running model.config.temperature and model.config.top_p.

Table 2. Model Versions (Table Composed on January 25, 2025)

Model	Hosted By	Model Size	Model Version
NTQAI/Nxcode-CQ-7B-orpo	Google Colab	7.25B	Model version as seen on https://huggingface.co/Artigenz/Artigenz-Coder-DS-6.7B/commits/main (from model card click on **Files and Versions** and then on **History: 7 commits** (3 commits may be different)): 74f3b3c06de36b261af9ef857279d6e33f893336, **commit of May 30, 2024**
Codestral Mamba	mistral.ai	7.3B	Endpoint: open-codestral-mamba. **Version: v 0.1**
Ministral 8B	mistral.ai	8B	Endpoint: ministral-8b-latest. **Version: 24.10**
deepseek-ai/deepseek-coder-6.7b-instruct	Google Colab		Model version as seen on https://huggingface.co/google/codegemma-7b-it/commits/main): e5d64addd26a6a1db0f9b863abf6ee3141936807, **commit of Feb 1, 2024**
Ministral 3B	mistral.ai	3B	Endpoint: ministral-3b-latest. **Version: 24.10**
Mistral-Nemo-Instruct-2407	mistral.ai	12B	Endpoint: open-mistral-nemo. **Version: 24.07**
meta-llama/Meta-Llama-3.1-8B-Instruct	Google Colab	8B	Model version as seen on https://huggingface.co/google/codegemma-7b-it/commits/main): 0e9e39f249a16976918f6564b8830bc894c89659, **commit of Sep 25, 2024**
Qwen/CodeQwen1.5-7B-Chat	Google Colab		Model version as seen on https://huggingface.co/google/codegemma-7b-it/commits/main): 7b0cc3380fe815e6f08fe2f80c03e05a8b1883d8, **commit of April 30, 2024**
m-a-p/OpenCodeInterpreter-DS-6.7B	Google Colab		Model version as seen on https://huggingface.co/google/codegemma-7b-it/commits/main): 60b89884df814590abd76757a6db4a527cbdfc91, **commit of Mar 3, 2024**
Mistral 7B	mistral.ai	7B	Endpoint: open-mistral-7b. **Version: v0.3**
Nous-hermes-2-solar-10.7b	replicate.com	10.7B	nateraw/nous-hermes-2-solar-10.7b:1e918ab6ffd5872c21fba21a511f344fd12ac0edff6302c9cd260395c7707ff4 (**1 year ago**)

Model	Hosted By	Model Size	Model Version
Phixtral-2x2_8 (4.5B)	replicate.com	4.5B	lucataco/phixtral-2x2_8:25d7b93bb0ec9e8dd94fcc69adc786759243a5628ba5574bd9609d6babafe57cf (**11 months, 2 weeks ago**)
Artigenz/Artigenz-Coder-DS-6.7B	Google Colab		Model version as seen on https://huggingface.co/google/codegemma-7b-it/commits/main): a0dea4a1c6cfdef8043c8accffa803887f444f45, **commit of April 16**
google/codegemma-7b-it	Google Colab	7B	Model version as seen on https://huggingface.co/google/codegemma-7b-it/commits/main): 078cdc51070553d1636d645c9a238f3b0914459a, **commit of Aug 7, 2024**
Slightly Bigger SLMs			
Mistral-Small-2409	mistral.ai	22B	Endpoint: mistral-small-latest. **Version: 24.09**
Codestral latest	mistral.ai	22.2B	Endpoint: codestral-latest. **Version: 25.01**
Mixtral-8x7B-v0.1	mistral.ai	12B active (47B total)	Endpoint: open-mixtral-8x7b. **Version: v0.1**
Not useful SLMs			
Qwen1.5-7b	replicate.com	7B	lucataco/qwen1.5-7b:f85bec5b21ba0860e0f200be6ef5af9d5a65b974b9f99e36eb036d21eab884de (**11 months, 2 weeks ago**)
Llama 3 8B	Replicate	8B	No version shown on replicate.com – hence the API error
Yi 6B (non-chat)	replicate.com	6B	01-ai/yi-6b:d302e64fad6b4d85d47b3d1ed569b06107504f5717ee1ec12136987bec1e94f1 (**1 year 2 months ago**)
Gemma 7B	replicate.com	7B	google-deepmind/gemma-7b:2ca65f463a2c0cfef4dbc4ba70d227ed96455ef6020c1f6983b2a4c4f3ecb4ec (**11 months ago**)
Gemma 2B	replicate.com	2B	google-deepmind/gemma-2b:26b2c530f16236a4816611509730c2e6f7b27875a6d33ec5cff42961750c98d8 (**11 months ago**)
Flan-T5	replicate.com		replicate/flan-t5-xl:eec2f71c986dfa3b7a5d842d22e1130550f015720966bec48beaae059b19ef4c (**1 year 9 months ago**)
Phi-2	replicate.com		lucataco/phi-2:740618b0c24c0ea4ce5f49fcfef02fcd0bdd6a9f1b0c5e7c02ad78e9b3b190a6 (**11 months, 3 weeks ago**)
Mamba 2.8B	replicate.com	2.8B	adirik/mamba-2.8b:571abd73203a3dd3d7071f1c0380a3502c427aba98a2fb5edf2f7cfdeea1676c (**11 months, 2 weeks ago**)

Source of model versioning information:

1. https://docs.mistral.ai/getting-started/models/models_overview/
2. https://replicate.com/explore
3. To determine versions for HuggingFace models, see the sha hash + date for the latest commit here: from **model card** click on **Files and versions,** then click on **History: 7 commits** (# commits may be different)

4. Analysis

- **Nxcode-CQ-7B-orpo** stands out as the top-scoring SLM overall.
- Other strong contenders include **Ministral 8B**, **Deepseek-Coder-6.7B, and Llama 3.1**.
- Several mid-tier models (e.g., CodeQwen1.5-7B-Chat, Mistral 12B Nemo, OpenCodeInterpreter) show moderate but inconsistent performance.
- Several models fall into low or nearly unusable categories due to extremely low pass rates or major practical limitations (long latencies, API errors, or nonsense outputs).
- **Larger parameter counts** do **not** always guarantee higher pass@1!

Top Performers

1. **Nxcode-CQ-7B-orpo**
   - *Human-Eval pass@1*: 82.93% / 87.23% (very high)
   - Solid MBPP (~73%) and LBPP (~22–24%)
   - Overall among the highest marks on multiple benchmarks.
2. **Ministral 8B**
   - *Human-Eval pass@1*: 72.56% / 76.8%
   - MBPP around 71%, mid-50% on other tasks, overall strong.
3. **Deepseek-Coder-6.7B-Instruct**
   - *Human-Eval pass@1*: 65.24% / 80.22%
   - Does well on MBPP (~70%) but struggles on LBPP (1%).
   - Not as consistently high as Nxcode, but still a strong contender.
4. **Ministral 3B** (with "instruct" variant)
   - *Human-Eval pass@1*: up to ~64.63% / 77.4% in instruct mode
   - Fairly good MBPP (~61–77%), decent LBPP (~20–27%).
   - Punches above its parameter count.
5. **Llama 3.1 8B Instruct**
   - *Human-Eval pass@1*: ~65.9% / 72.6%
   - MBPP and LBPP in the mid 50–60% range, a respectable showing.

These top models generally exceed ~60% pass@1 on Human-Eval (sometimes well above 70–80%), plus moderate to good results on MBPP and other code tasks.

Mid Performers

1. **CodeQwen1.5-7B-Chat**

    - *Human-Eval pass@1*: 50% / 87.2% (unclear if the 87.2% is a different setting)
    - MBPP ~55%, LBPP ~19–26%.
    - Results are somewhat mixed but places it in a middle tier on average.
2. **Mistral-Nemo-Instruct-2407 (12B)**
    - *Human-Eval pass@1*: ~58.5% / 67%
    - MBPP ~47%, LBPP ~21–22%.
    - Decent but not top-tier.
3. **OpenCodeInterpreter-DS-6.7B**
    - *Human-Eval pass@1*: ~41% / 73.2%
    - Good MBPP (~72%), but quite low performance on some tasks like LBPP (5–8%).
4. **Mistral 7B** (open-mistral-7b)
    - *Human-Eval pass@1*: ~31%
    - MBPP ~13.6%, LBPP ~9.9%.
    - Sits lower than the ones above, but still not in the "near-zero" group.

Many of these mid-range models have partial strong points (e.g., decent MBPP or decent instruct performance) but are inconsistent across benchmarks.

Low Performers

A number of models show **very low** pass@1 on Human-Eval (often near 0–25%) or produce mostly irrelevant outputs:

- **Nous-hermes-2-solar-10.7b** (~25.6% Human-Eval)
- **Phixtral-2x2_8 (4.5B)** (~14.64%)
- **Artigenz-Coder-DS-6.7B** (1.22% / 70.89% in some mode, but near 0% in others)
- **Code Gemma 7b IT** (0% on some tasks)

And several models from the "Not Useful SLMs" section with near-zero performance or major usability problems (API errors, extremely long latencies, or nonsense outputs):

- **Qwen1.5-7B** (unusable due to 200 s API calls)
- **Llama 3 8B (Replicate)** (API errors)
- **Yi 6B**, **Gemma 7B**, **Gemma 2B**, **Flan-T5**, **Phi**, **Mamba 2.8B** all show 0–3% pass@1 or produce irrelevant outputs.

5. Summary of what was done

- Table contains much more data now.
- Finished the **first and second HumanEval runs and the MBPP run** – CONSIDERABLE EFFORT as the dataset has 500 data points which means the code needs to be generated and verified 500 times.
- According to (Matton et al. 2024), **data leakage** in code generation occurs when popular evaluation benchmarks (like HumanEval and MBPP) appear in a model's training data and, whether intentionally or unintentionally, compromise the validity of test scores. Therefore, two more runs were done on the **LBPP** and **Big Code Bench** dataset for all models.

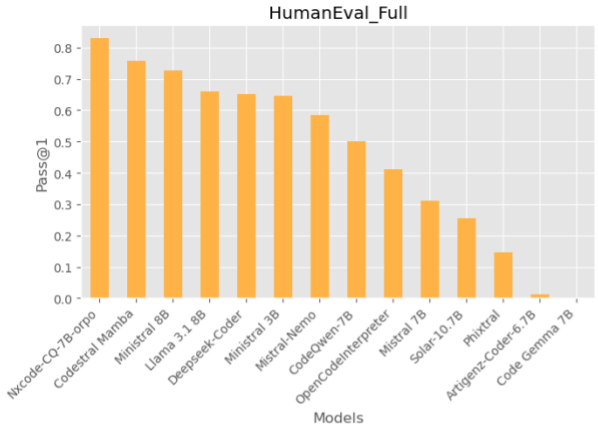
- Conducted an initial experiment with the **Reflection agentic workflow**.
- Need to continue applying various agentic workflows.
- Need to finish the **Methodology** section

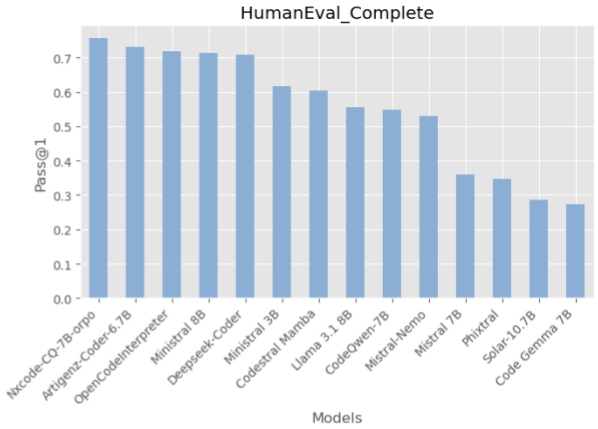
References

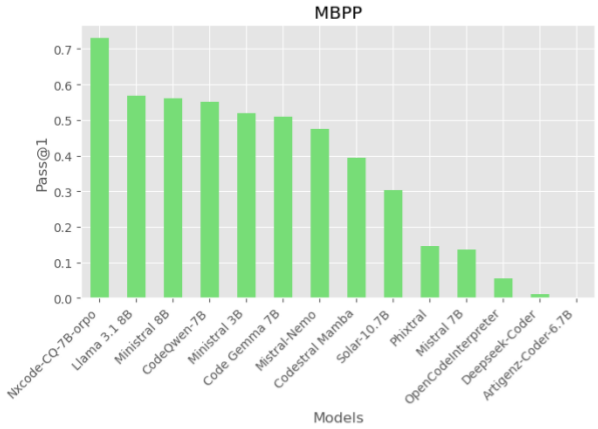
Matton A., Tom Sherborne, Dennis Aumiller, Elena Tommasone, Milad Alizadeh, Jingyi He, Raymond Ma, Maxime Voisin, Ellen Gilsenan-McMahon, Matthias Gallé. 2024. **On Leakage of Code Generation Evaluation Datasets**.

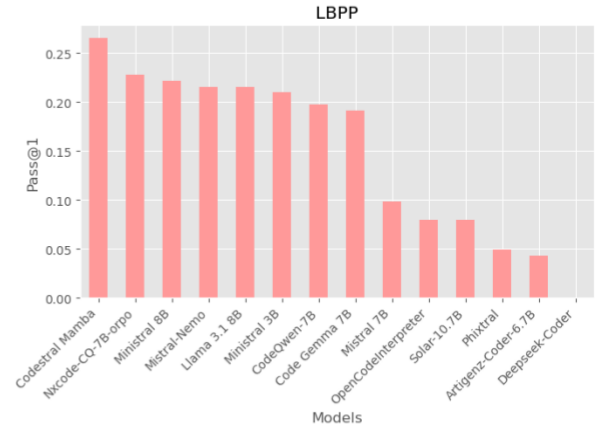
Visuals

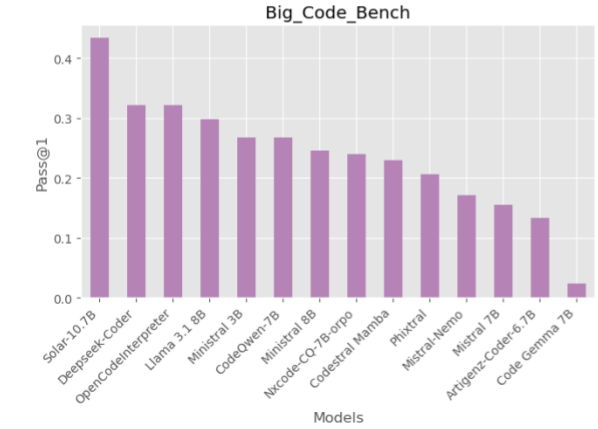
1. Pass@1 Scores Visualized by Model / Dataset











2. Initial Results for Models by Dataset

	Model	HumanEval_Full	HumanEval_Complete	MBPP	LBPP	Big_Code_Bench
0	Nxcode-CQ-7B-orpo	0.8293	0.75610	0.730	0.2284	0.240
1	Codestral Mamba	0.7561	0.60370	0.394	0.2654	0.230
2	Ministral 8B	0.7256	0.71340	0.562	0.2222	0.246
3	Deepseek-Coder	0.6524	0.70730	0.010	0.0000	0.322
4	Ministral 3B	0.6463	0.61590	0.518	0.2099	0.268
5	Mistral-Nemo	0.5854	0.53050	0.474	0.2160	0.172
6	Llama 3.1 8B	0.6590	0.55470	0.568	0.2160	0.298
7	CodeQwen-7B	0.5000	0.54880	0.552	0.1975	0.268
8	OpenCodeInterpreter	0.4100	0.71950	0.054	0.0800	0.322
9	Mistral 7B	0.3110	0.35980	0.136	0.0990	0.156
10	Artigenz-Coder-6.7B	0.0122	0.73170	0.002	0.0432	0.134
11	Code Gemma 7B	0.0000	0.27440	0.510	0.1914	0.024
12	Solar-10.7B	0.2561	0.28650	0.304	0.0800	0.434
13	Phixtral	0.1464	0.34756	0.146	0.0490	0.206

3. Normalized Results with Final Ranking

	Model	HumanEval_Full	HumanEval_Complete	MBPP	LBPP	Big_Code_Bench	Average
0	Nxcode-CQ-7B-orpo	1.000000	1.000000	1.000000	0.860588	0.526829	0.877483
2	Ministral 8B	0.874955	0.911356	0.769231	0.837227	0.541463	0.786846
1	Codestral Mamba	0.911733	0.683621	0.538462	1.000000	0.502439	0.727251
6	Llama 3.1 8B	0.794646	0.581897	0.777473	0.813866	0.668293	0.727235
4	Ministral 3B	0.779332	0.708947	0.708791	0.790882	0.595122	0.716615
7	CodeQwen-7B	0.602918	0.569649	0.755495	0.744160	0.595122	0.653469
5	Mistral-Nemo	0.705897	0.531659	0.648352	0.813866	0.360976	0.612150
8	OpenCodeInterpreter	0.494393	0.924019	0.071429	0.301432	0.726829	0.503620
3	Deepseek-Coder	0.786688	0.898692	0.010989	0.000000	0.726829	0.484640
12	Solar-10.7B	0.308815	0.025119	0.414835	0.301432	1.000000	0.410040
9	Mistral 7B	0.375015	0.177289	0.184066	0.373022	0.321951	0.286269
11	Code Gemma 7B	0.000000	0.000000	0.697802	0.721176	0.000000	0.283796
10	Artigenz-Coder-6.7B	0.014711	0.949346	0.000000	0.162773	0.268293	0.279025
13	Phixtral	0.176534	0.151879	0.197802	0.184627	0.443902	0.230949

See the **Average** column for final ranks (after min max scaling).

Next Steps

(This was not submitted on January 25 – for my reference only)

1. Finish the **temperature & top_k** experiments – decide which models to discontinue (Phixtral and Artigenx? Or more?).
2. Finalize the format of the **Reflection workflow**: keep simple as I did with Llama 3.1 or use Langchain or another format?
3. Run Reflection workflow for **all models all datasets**.
4. Do 1 experiment with **another agentic workflow** (Llama?)
5. Finish the Chapter 3 **Methodology**.

Other feedback from Dr. A.:

Show the difference in the boost for other models, how much uplift when using agents. Have the delta in performance as a separate column. Maybe smaller models have a bigger boost which is even more important for the companies that want to use this at scale? (20% instead of 12%) For example, some companies may want to offer code generation as a service to non-technical companies (my comment – already happening with Co-Pilot and similar products).

For defense – prepare a ppt that runs through Chapters 1 through 5. Will not be reviews by Dr. A. – to demo how capable a doctoral student is of incorporating feedback and updating the results (because next the student will lead people in the AI domain). IMPORTANT: the student will be cut off after 30 min – be very mindful on how much you can fit in because you don't want to be cut off in the Methodology section without even presenting the results. Optimal ppt length – 25 minutes. Practice speaking through it.