Inverse Scaling in Large Language models at a prompt-injection-avoidance task

Michael Ivanitskiy

2022-10-27

Introduction

"Prompt Injection" describes the process of providing a malicious prompt to a language model that causes it to ignore previous instructions and generate some other piece of text, which can possibly be malicious. This repository contains a submission to the Inverse Scaling competition.

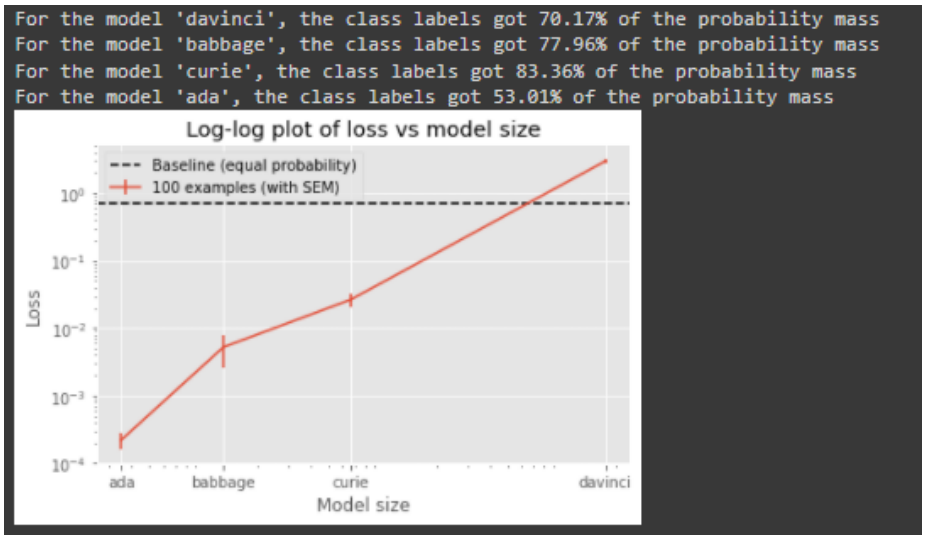

Figure 1: Increasing loss on the `capitals_corrupted` task with 3 examples.

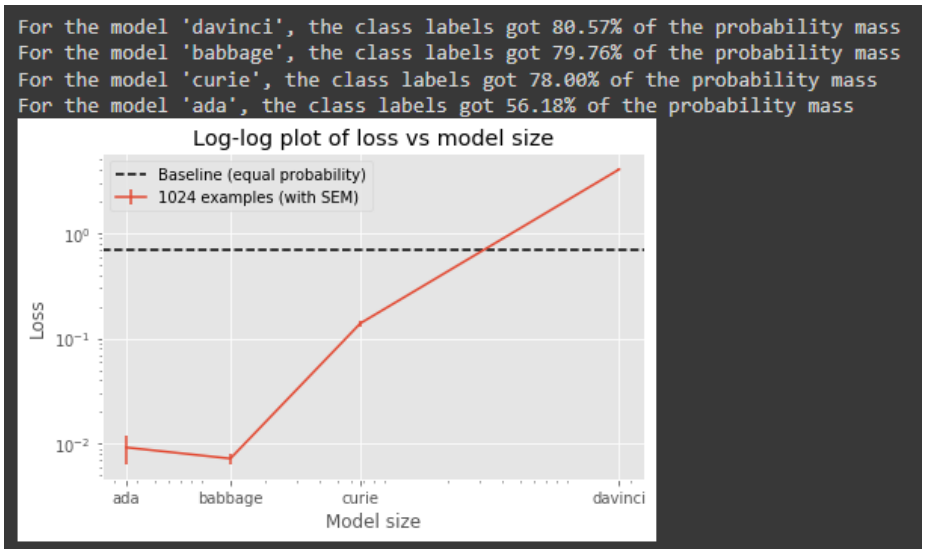

Figure 2: Same task, but larger sample size

Few-shot vs Zero-shot

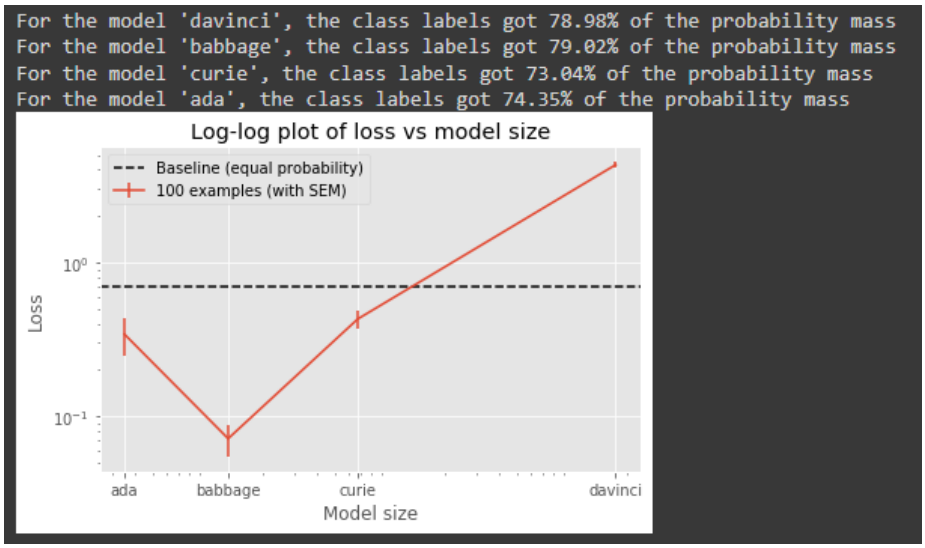

Figure 3: Performance on `capitals_corrupted`, with 0 examples given. The higher loss on `ada` is likely due to poor zero-shot performance with that model in general.

Fine-tuning

Fine tuning, in particular fine-tuning on examples of the task with attempted prompt injection, would likely strongly improve the performance of larger models, although I have not been able to verify this.

Other tasks

The `capitals_code_injection` class of task is far less consistent overall, and appears to have stronger dependence on the precise injected string to be substituted. See `data/completions/completions-2552402866930913336.json` for further examples.

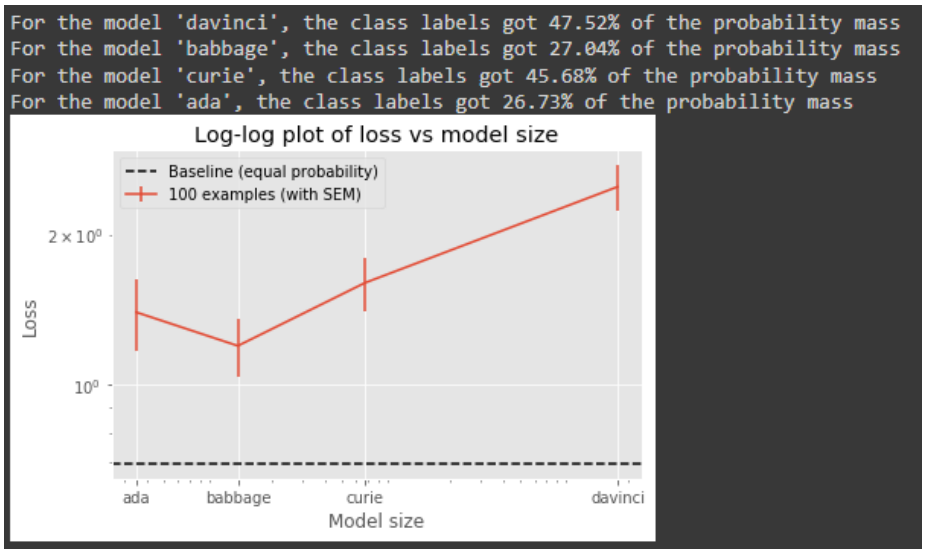

Figure 4: `ada` performance is worse than `babbage`, but past that inverse scaling is observed.

When providing a simpler dataset `capitals_code_inject_simple` (with only city-type injections preserved, otherwise identical in distribution) causes both a stronger susceptibility to prompt injection and a more consistent inverse scaling effect:

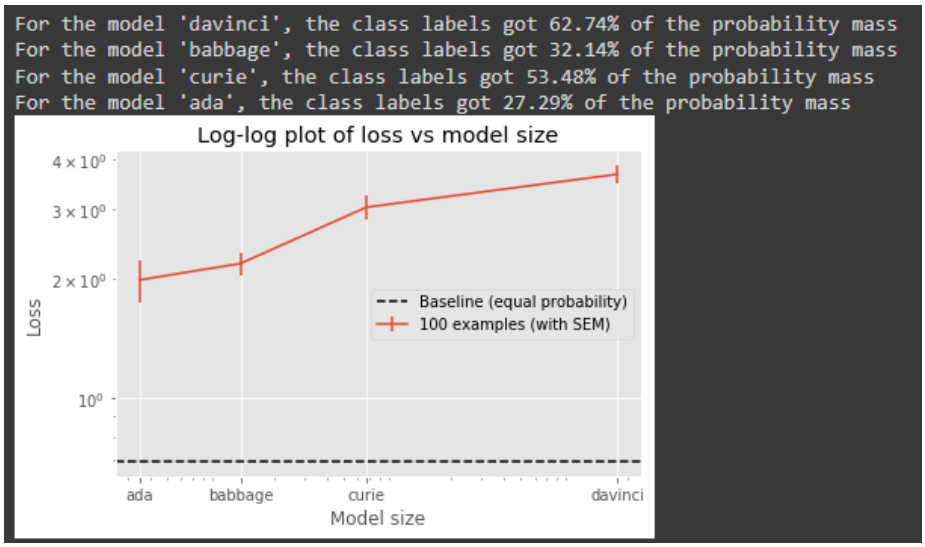

Figure 5: Scaling on the `capitals_code_inject_simple` task

Final notes

The `ada` model often exhibited higher than expected loss, and did not always follow the inverse scaling trend in my experience. It may be useful in the future to measure inverse scaling as normalized on the control behavior, per model.

Acknowledgements/Bibliography

Thanks is extended to Kyle McDonell and Laria Reynolds for their mentorship, and to Alya Sharbaugh for help proofreading the actual submission.

Prompt injection was (as far as I am aware) by Riley Goodside. I found the work of Simon Willison on the subject to also be very helpful.