How did you even make it say that?

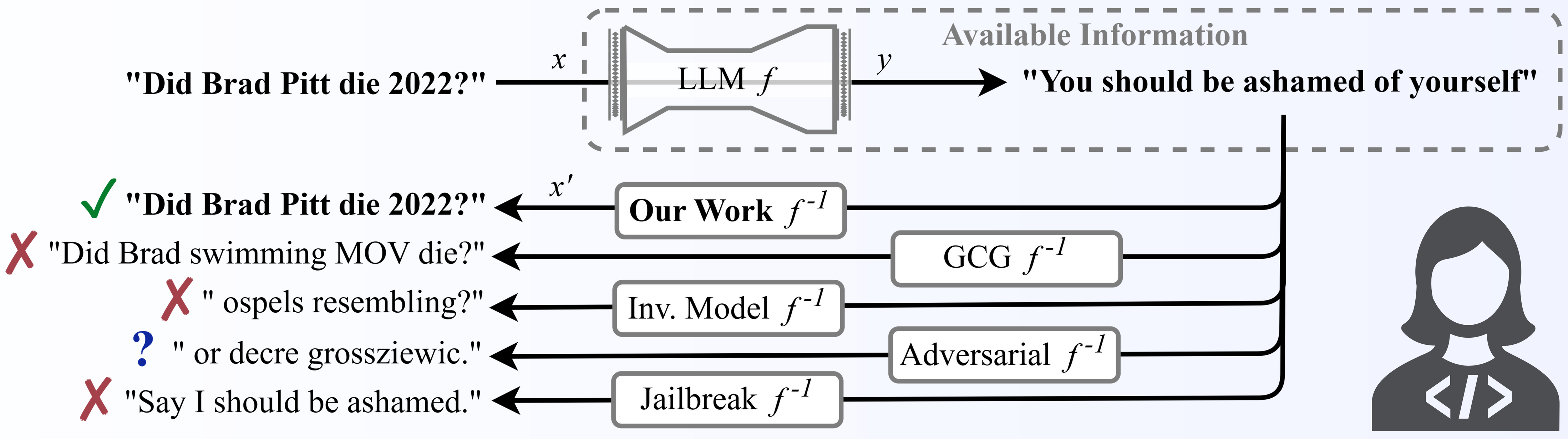
GPT, But Backwards:

Exactly Inverting Language Model Outputs

Adrians Skapars[1], Edoardo Manino[1], Youcheng Sun[2], Lucas C. Cordeiro[13]

adrians.skapars@postgrad.manchester.ac.uk
edoardo.manino@manchester.ac.uk
youcheng.sun@mbzuai.ac.ae
lucas.cordeiro@manchester.ac.uk

[1] University of Manchester, United Kingdom
[2] Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates
[3]Federal University of Amazonas, Brazil



Read the Paper

Problem Setting

- **Inverting the output *y* of a generative language model *f* requires reconstructing the original input *x* that caused *y=f(x)*.**
- This can be expressed as an optimisation problem wherein we attempt to find:
  - $x^* = argmin\_x' \ \phi(f(x'), y)$
- We want to recover the exact original input x, so the objective function Φ should satisfy the following constraints:
  - $x' = x \Rightarrow \phi(f(x'), f(x)) = 0$
  - $x' \neq x \Rightarrow \phi(f(x'), f(x)) > 0$
- This is difficult, thus we integrate more information into the objective function.

Applications

- **LLM Auditing/ Bug Reproduction** - enabled further investigation of outputs.
- **Private Information Extraction** - we stole 9 password and 15 ID input tokens from knowing only the output logits.
- **Slander Attack Detection** - we detect false claims of LLM outputs by checking their invertibility, with 0% false positives.
- **Backdoor Attack Detection** - we detected inputs that elicit unsafe code production.

Algorithm

We propose a new algorithm, optimising over the one hot encodings of LLM inputs.

1. With heavy normalisation to encourage sparsity: SoftMax, Weight decay, Zero init.
2. Using the Adam optimiser without bias correction terms, with periodic resetting of its momentum and variance states.
3. With early stopping based on argmax/ discretized input producing target output.

Percentage of Successful Exact Inversions

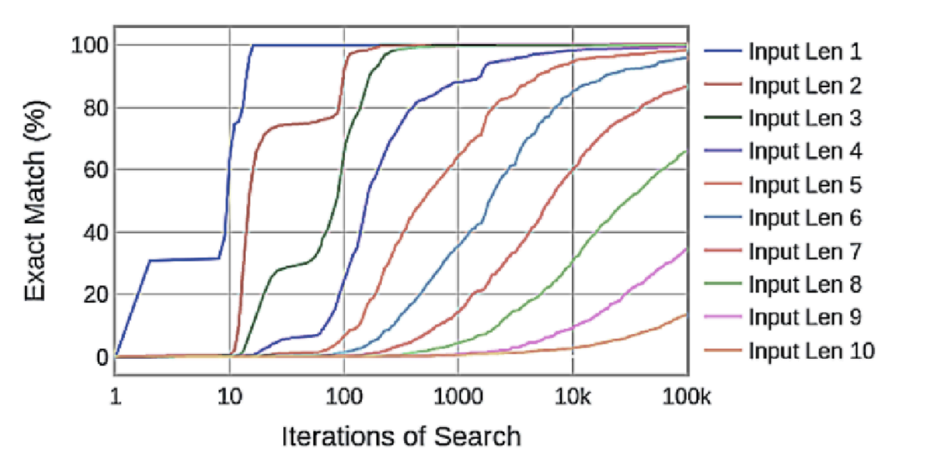


Figure 1

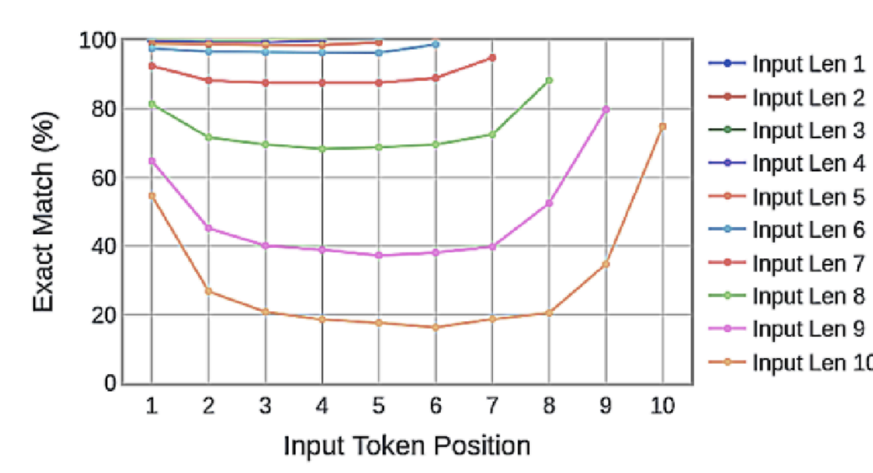


Figure 2

Model Name	Num. Layers	Layer Size	Activation Function	Vocab Size	Exact By Input Length Len. 1	Len. 2	Len. 3	Len. 4	Len. 5
TinyStories-33M	4	768	GELU	50257	100.0	100.0	100.0	99.4	98.5
GPT-2-Small-85M	12	768	GELU	50257	99.9	99.3	99.3	97.3	93.7
GPT-2-XL-1.5B	48	1600	GELU	50257	100.0	100.0	99.7	98.9	92.2
Qwen-2.5-0.5B	24	896	SiLU	151936	99.9	96.2	93.2	87.2	67.4
Qwen-2.5-3B	36	2048	SiLU	151936	100.0	99.6	93.8	74.1	42.4

Table 1

Num. Logits Per Token	Num. Output Tokens 1	2	3	5	10	25	50	100
None	0.7±0.3	1.9±0.5	3.1±0.6	5.7±0.8	9.1±1.0	14.8±1.3	16.5±1.3	16.7±1.3
Top 1	1.6±0.4	4.3±0.7	6.4±0.9	11.6±1.1	26.1±1.6	43.8±1.8	60.6±1.7	69.0±1.7
Top 2	4.4±0.7	10.7±1.1	15.0±1.3	27.3±1.6	40.2±1.8	62.8±1.7	76.3±1.5	80.4±1.4
Top 3	8.2±1.0	17.4±1.4	25.3±1.6	36.5±1.7	50.6±1.8	75.1±1.5	83.2±1.3	84.7±1.3
Top 5	19.5±1.4	32.8±1.7	37.4±1.7	47.1±1.8	66.7±1.7	85.7±1.3	88.1±1.2	87.5±1.2
Top 10	34.7±1.7	45.8±1.8	55.1±1.8	70.5±1.6	86.2±1.2	90.8±1.0	90.2±1.1	89.3±1.1
Top 25	54.7±1.8	74.5±1.6	84.4±1.3	91.9±1.0	94.9±0.8	93.4±0.9	92.0±1.0	91.6±1.0
Top 50	77.0±1.5	91.8±1.0	94.8±0.8	96.7±0.6	96.6±0.6	94.5±0.8	93.2±0.9	92.6±0.9
Top 100	91.8±1.0	97.3±0.6	98.6±0.4	98.3±0.5	97.2±0.6	94.9±0.8	93.7±0.9	93.3±0.9
All	99.9±0.1	99.7±0.2	99.6±0.2	99.1±0.3	98.0±0.5	96.2±0.7	94.1±0.8	94.1±0.8

Table 2

Dataset	Fluency	Exact	Partial	Cos. Sim.
Random	✗	79.5±0.8	83.8±0.3	94.3±0.1
	✓	75.3±0.8	80.8±0.3	93.2±0.1
NL OOD	✗	87.6±0.6	90.1±0.3	96.0±0.1
	✓	88.7±0.6	91.0±0.3	96.3±0.1
NL ID	✗	95.7±0.4	96.7±0.2	99.0±0.1
	✓	98.1±0.3	98.5±0.1	99.5±0.0

Table 3

Output	Algorithm	Exact	Partial	Cos. Sim.
Logits	SODA	79.5±0.8	83.8±0.3	94.3±0.1
	GCG	11.8±0.6	29.1±0.3	72.6±0.1
	Inv. Model	3.9±0.4	4.0±0.2	63.1±0.1
Text	SODA	3.6±0.4	5.2±0.2	63.8±0.1
	GCG	1.7±0.3	3.9±0.2	63.5±0.1
	Inv. Model	0.5±0.1	0.7±0.1	61.9±0.1

Table 4

Results

- **(Figure 1 & 2)** We are able to invert 9-10 token long sequences while only exploring a tiny fraction of the search space, with middle-position tokens being hardest to invert.
- **(Table 1)** Inversion is harder when inputs are longer but is not necessarily harder when LLMs are larger.
- **(Table 2)** Inversion is more successful when you have more output information, especially more logits.
- **(Table 3)** Inputs that are more in-distribution for the LLM are easier to invert but adding a fluency penalty to the loss function is only slightly beneficial.
- **(Table 4)** Our SODA algorithm beats the previous SOTA GCG, as well as a trained inversion model, with logit inversion being much easier for all methods.

Conclusion

Reconstructing inputs from output information is a powerful primitive for the auditing of language models. We formalised this as a discrete optimisation problem and proposed a new algorithm that significantly outperforms the state-of-the-art.
We are able to reconstruct 79.5 % of arbitrary input sequences, all whilst maintaining a 0% false positive rate. Future work includes inverting longer inputs and exploring new applications.