A Scientific approach to Transformers

A scientific approach to transformers is a prerequisite for anyone who wishes to implement an AI solution seriously. This notebook questions the assertions made regarding Macaw. This notebook questions the following statement by AllenNLP: "On a suite of 300 challenge questions. Macaw outperformed GPT-3 by over 10%, even though Macaw is an order of magnitude smaller (11 billion vs. 175 billion parameters)."

As this notebook shows, if an expert runs a GPT-3 engine, the accuracy is extremely high. Why? Because GPT-3 is a model, NOT an engine. You have to select the right GPT-3 engine and parameters and be an expert in prompt engineering. I ran several examples that contradict the results displayed on AllenNLP's website using a Hugging Face T5 model.

You cannot assert that a tool is THE best before you apply a scientific approach to the problem at hand. OpenAI, Hugging Face, and AllenNLP are tools. Do you know which one to use, when to use it and how to implement it before a deep investigation?

No, you don't.

In Chapter 1 of Transformers for NLP, 2nd Edition,2022, I explain why you cannot start by saying "Hugging Face" or "AllenNLP," or "OpenAI" is THE best solution. It's like choosing a tool before knowing what you need one for!

AI2 Allen Institute for AI

Macaw

Macaw is a high-performance question-answering (QA) model capable of outperforming other popular current language models, all while being an order of magnitude smaller. This demo allows you to explore Macaw's answers and compare them to those of the popular GPT-3 language model on a benchmark set of questions.

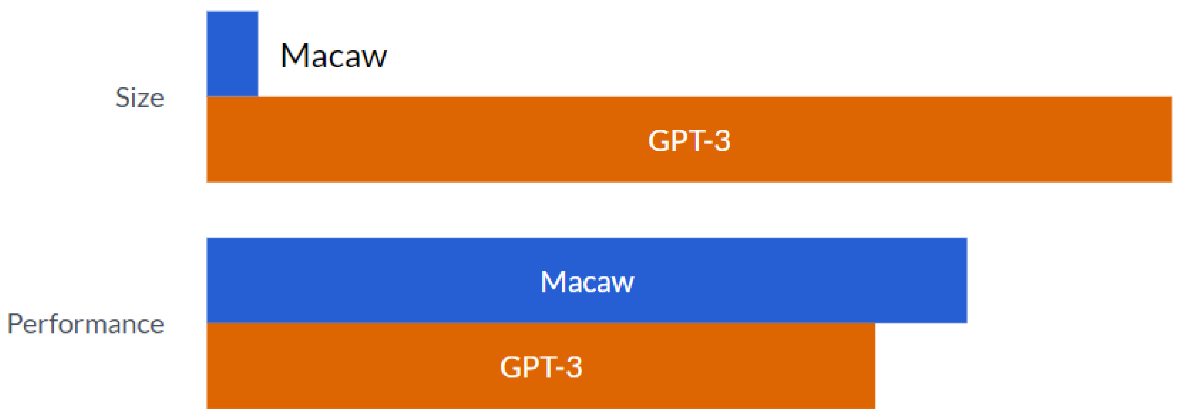
Choose one or more question categories:

All

Filter questions by:

All

Model Size vs Model Performance



Answer key:

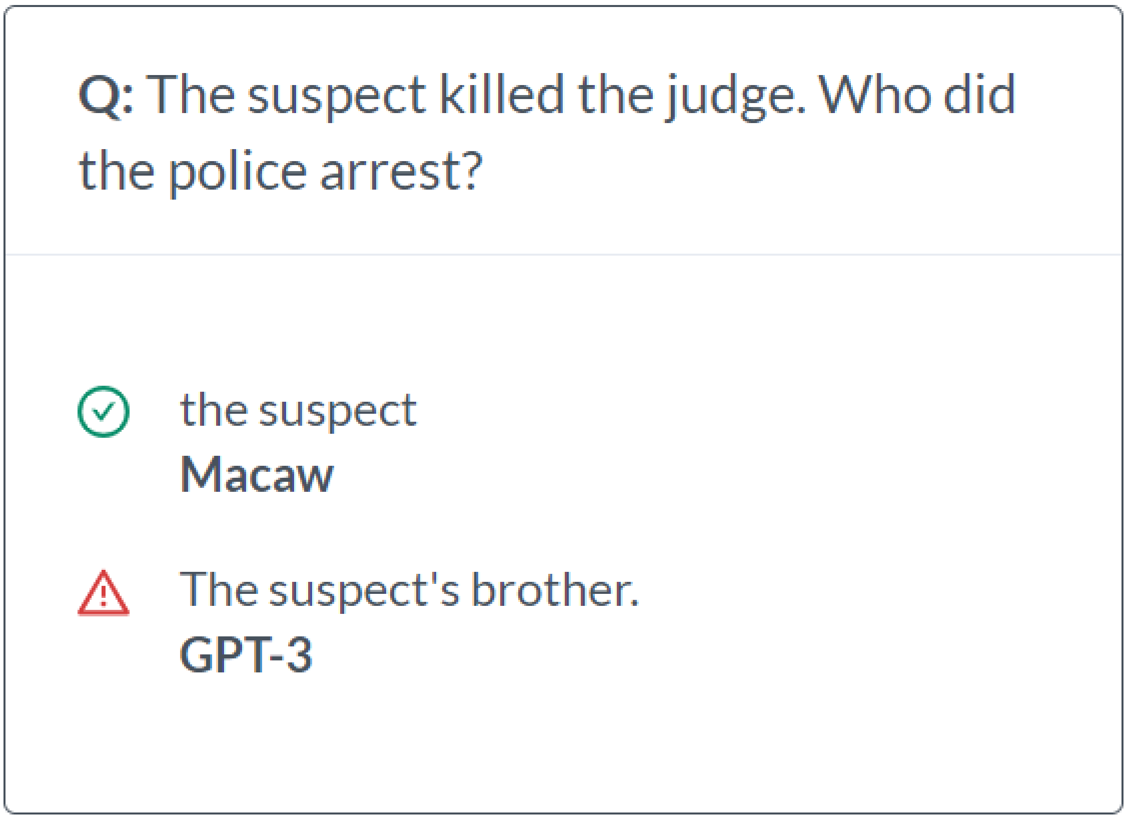
✓ Correct	! Partially correct	⚠ Incorrect

```
!pip install openai
```

```
import openai
```

```
f = open("api_key.txt", "r")
API_KEY=f.readline()
f.close()
```

```
#The OpenAI Key
import os
os.environ['OPENAI_API_KEY'] =API_KEY
openai.api_key = os.getenv("OPENAI_API_KEY")
```


Q: The suspect killed the judge. Who did the police arrest?

✓ the suspect
Macaw

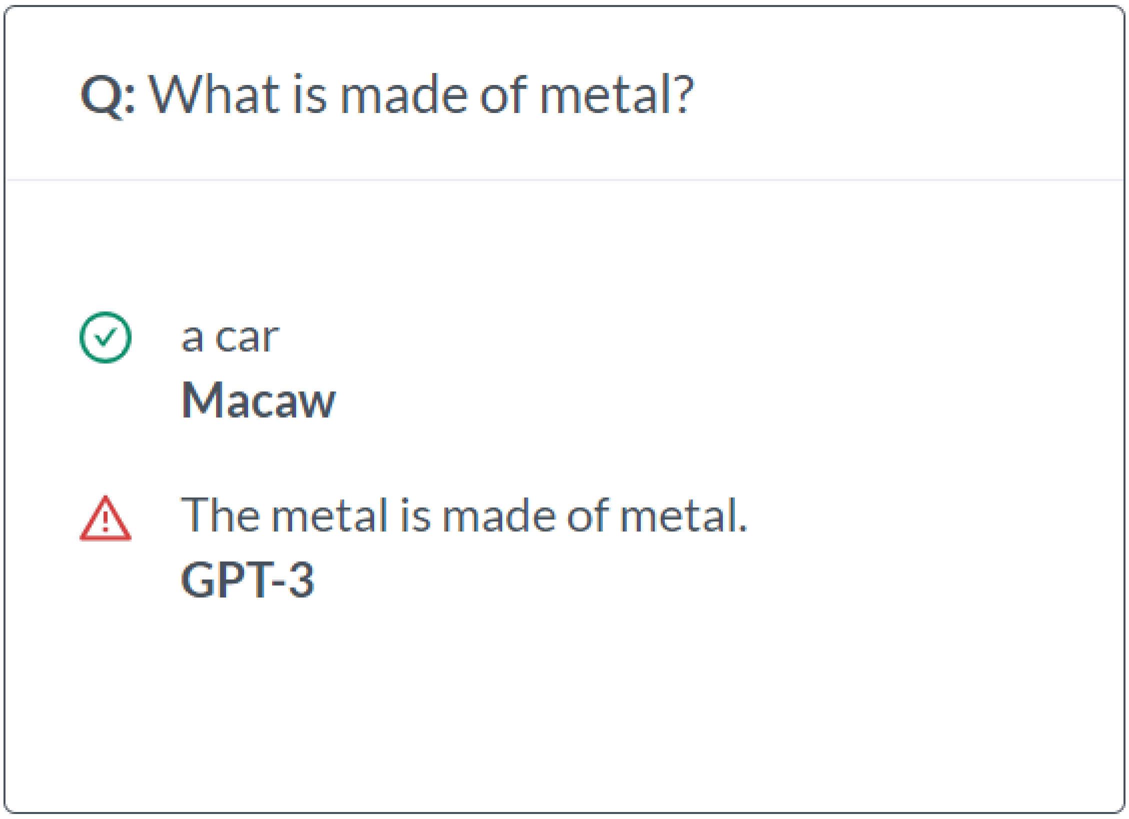
⚠ The suspect's brother.
GPT-3

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="Q:The judge was killed by the suspect. Who did the police arrest?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

```
 The police arrested the suspect.
```

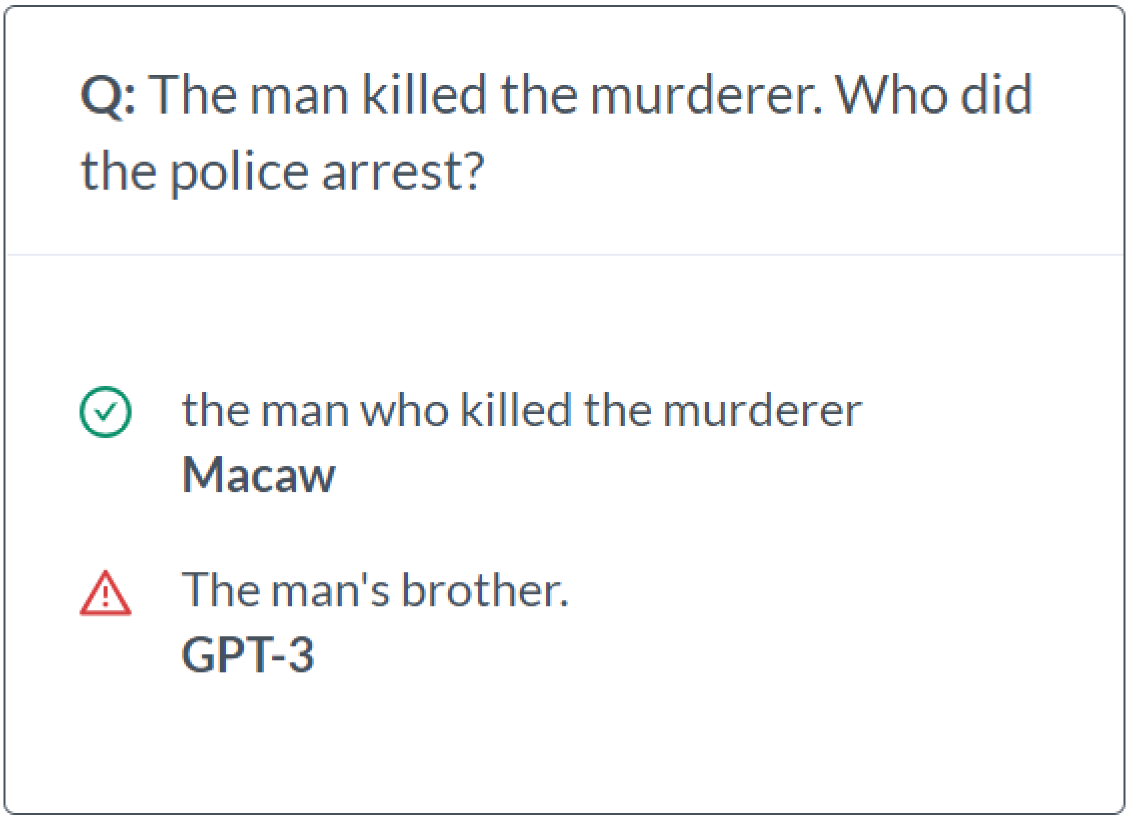
The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.



Q: What is made of metal?

✓ a car
Macaw

⚠ The metal is made of metal.
GPT-3



Q: The man killed the murderer. Who did the police arrest?

✓ the man who killed the murderer
Macaw

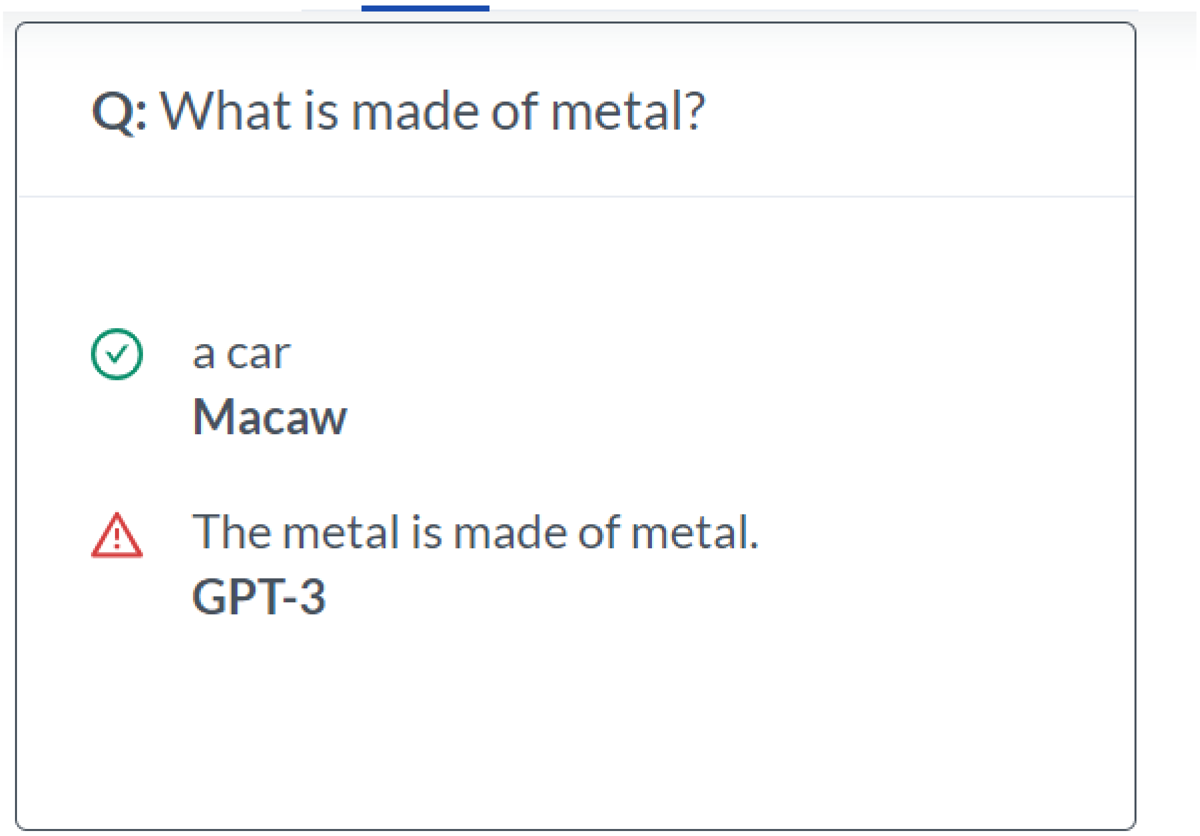
⚠ The man's brother.
GPT-3

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="The man killed the murderer. Who did the police arrest?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

```
 The man killed the murderer. The police arrested the man.
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.



Q: What is made of metal?

✓ a car
Macaw

⚠ The metal is made of metal.
GPT-3

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="What is made of metal?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

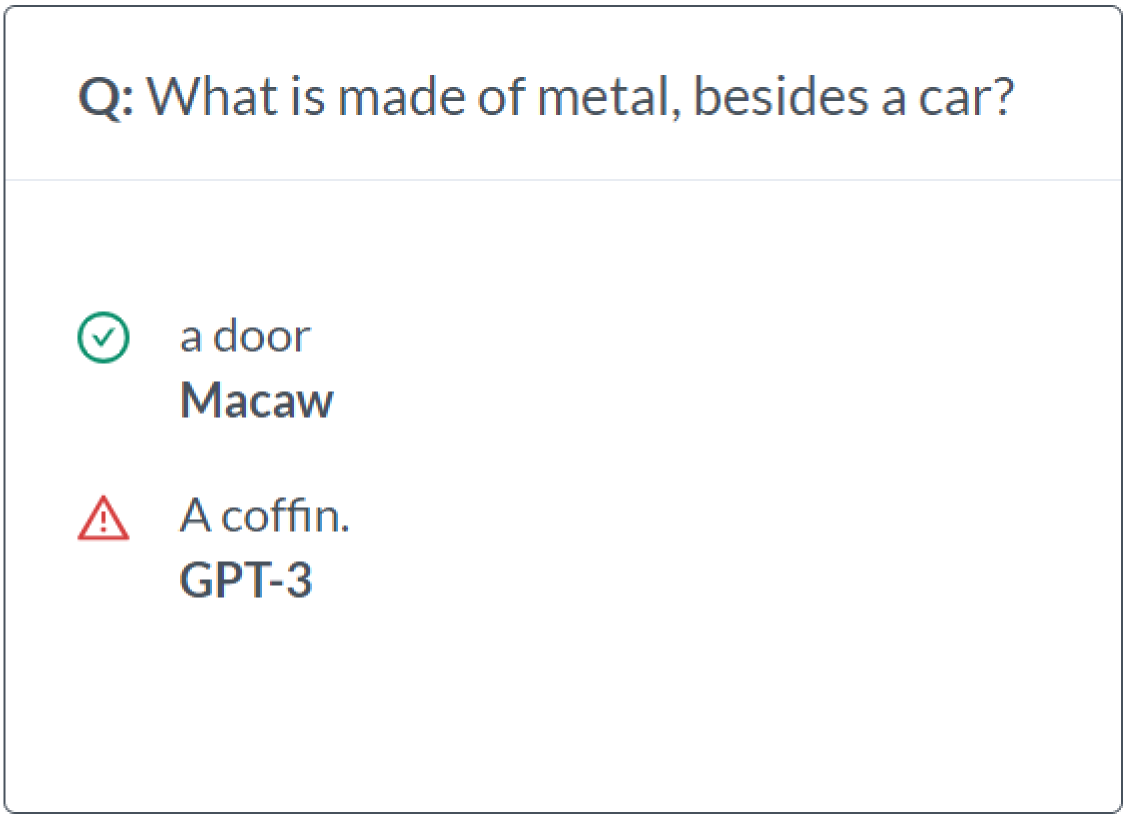
```
Metal is made of different types of elements, including iron, copper, and aluminum.
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="What things are made of metal?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

```
Metal is a material that is typically hard, shiny, and has a good electrical and the
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.



Q: What is made of metal, besides a car?

✓ a door
Macaw

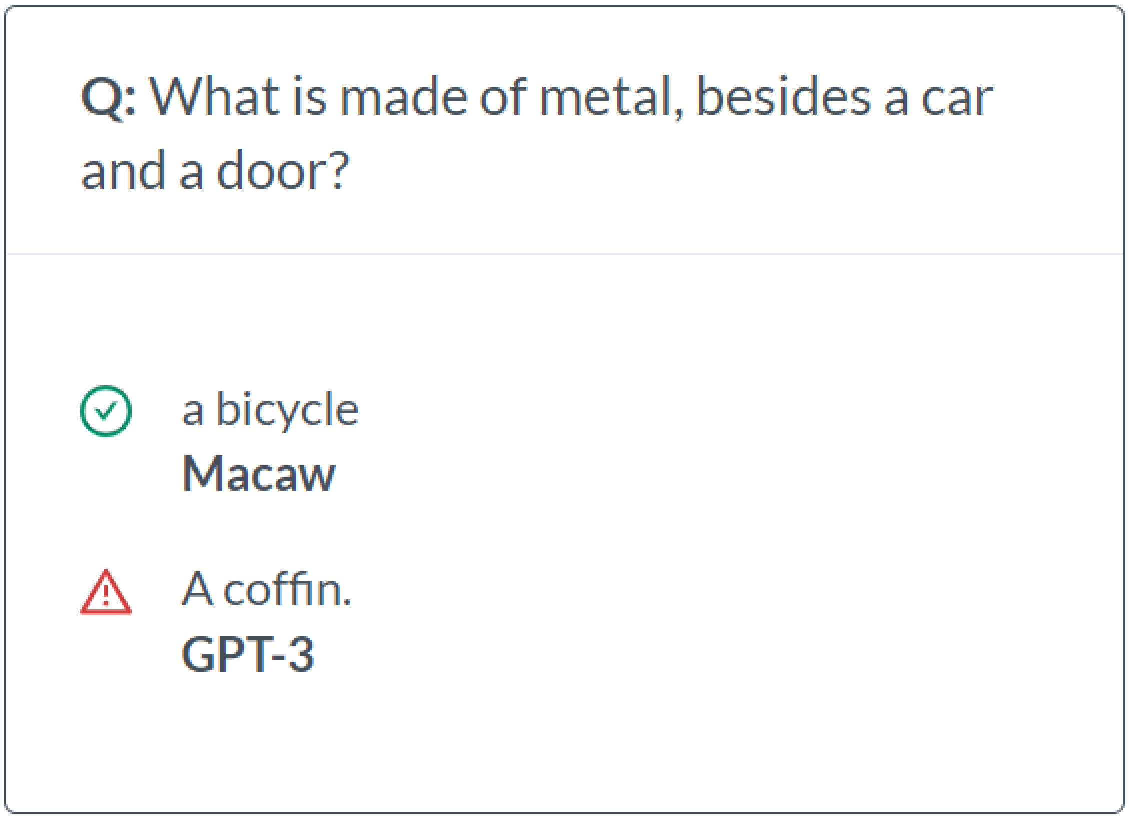
⚠ A coffin.
GPT-3

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="What is made of metal, besides a car?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

```
     A bike.
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.



Q: What is made of metal, besides a car and a door?

✓ a bicycle
Macaw

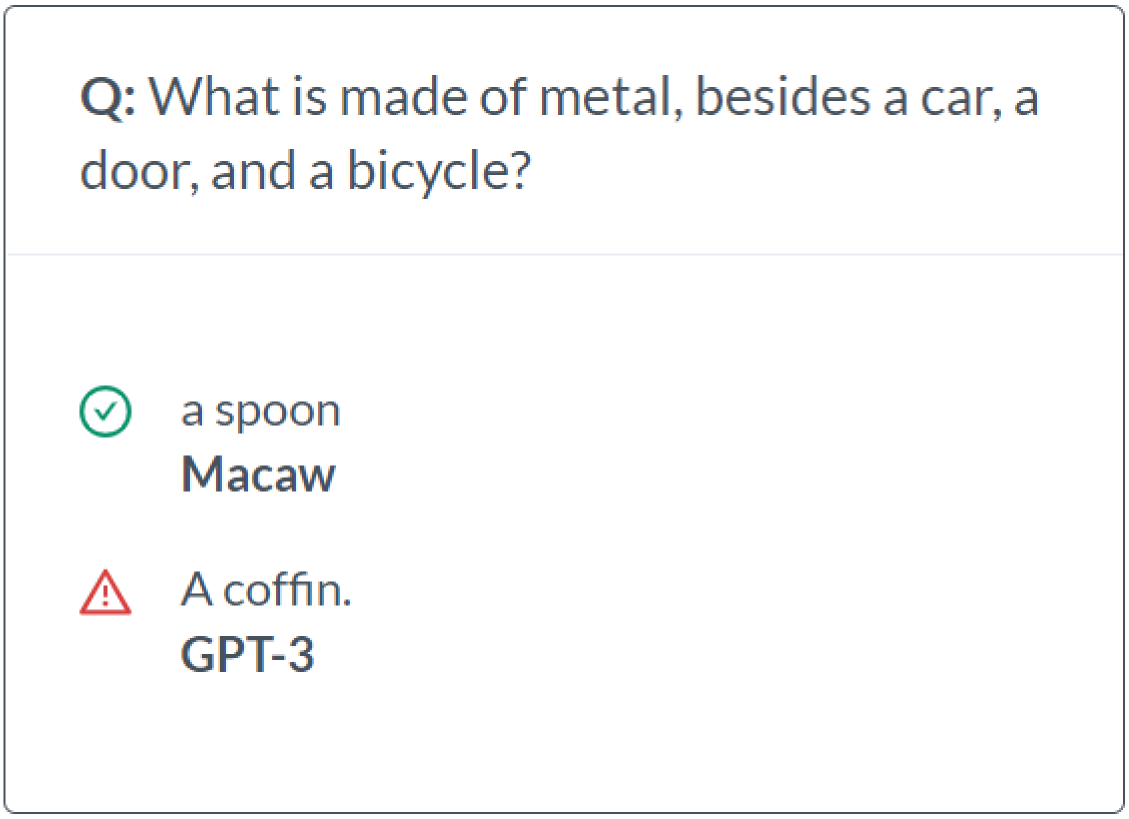
⚠ A coffin.
GPT-3

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="Q: What is made of metal, besides a car and a door?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

```
 A can opener.
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.



Q: What is made of metal, besides a car, a door, and a bicycle?

✓ a spoon
Macaw

⚠ A coffin.
GPT-3

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="Q: What is made of metal, besides a ca, a door, and a can opener?\n\nA:",
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
```

```
r = (response["choices"][0])
print(r["text"])
```

```
    A sink.
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.

More Macaw Examples

Is it true that GPT-3 cannot provide the correct output? Let's see for ourselves.

```
response = openai.Completion.create(
  engine="text-davinci-001",
  prompt="Q:James went camping in the woods, but forgot to bring a hammer to bang the t
  temperature=0.7,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0,
  stop=["\n"]
)
r = (response["choices"][0])
print(r["text"])
```

```
    He could use a rock to bang the tent pegs in.
```

The GPT-3 text-davinci-001 engine (GPT-3 is a model, not an engine) with the correct parameters produced the correct output.

Scientific Conclusion

1. The OpenAI platform works fine if you know when and how to use it.
2. The Hugging Face platform works fine if you know when and how to use it.
3. The AllenNLP platform works fine if you know when and how to use it.

So, in the end, the core problem is to find the right tool with a scientific method. And that is something you cannot assert in advance, as I explain in Transformers for NLP, 2nd Edition, 2022 starting chapter 1.

My guiding principle: You can use AI as a tool or be the tool of AI.

Important warning: Transformers are STOCHASTIC models. You may not obtain the same result each time you run one. That's inherent to AI models that are NOT rule-based.

References

OpenAI with excellent tools if you know when and how to use them.

Hugging Face with excellent tools if you know when and how to use them.

AllenNLP with excellent tools if you know when and how to use them.

Transformers for NLP, 2nd Edition, 2022 that provides you with insights on OpenAI, Hugging Face, AllenNLP, and more to make you THE expert that knows how to choose the right tool at the right time for the right task.