Model Card (Orca-Mistral)

Link: microsoft/Orca-Mistral · Hugging Face

Orca-Mistral is built for research purposes only and provides a single turn response in tasks such as reasoning over user given data, reading comprehension, math problem solving and text summarization. The model is designed to excel particularly in reasoning.

Note that:

1. This is a research model, intended to show that we can use capable models and complex workflows (advanced prompts, multiple calls) to create synthetic data that can teach Small Language Models (SLMs) new capabilities. We chose reasoning because it is a widely useful capability that SLMs lack.
2. The model is not optimized for chat and has not been trained with RLHF or DPO. It is best used after being finetuned for chat or for a specific task.
3. Beyond reasoning, the model inherits capabilities and limitations of its base (Mistral-7b). We have already seen that the benefits of the Orca training can be applied to other base model too.

We make Orca-Mistral's weights publicly available to support further research on the development, evaluation, and alignment of SLMs.

What is Orca-Mistral's intended use(s)?

- Orca-Mistral is built for research purposes only.
- The main purpose is to allow the research community to assess its abilities and to provide a foundation for building better frontier models.

How was Orca-Mistral evaluated? [TODO: Add main eval result]

- Orca-Mistral has been evaluated on a large number of tasks ranging from reasoning to grounding and safety.

Model Details

Orca-Mistral is a finetuned version of Mistral-7b. Orca-Mistral's training data is a synthetic dataset that was created to enhance the small model's reasoning abilities. All synthetic training data was moderated using the Microsoft Azure content filters. More details about the model can be found in the Orca 2 paper.

Please refer to Mistral-7b technical report for details on the model architecture.

License

Orca-Mistral is licensed under the MIT License.

Bias, Risks, and Limitations

Orca-Mistral, built upon the Mistral-7b, retains many of its limitations, as well as the common limitations of other large language models or limitation caused by its training process, including:

Data Biases: Large language models, trained on extensive data, can inadvertently carry biases present in the source data. Consequently, the models may generate outputs that could be potentially biased or unfair.

Lack of Contextual Understanding: Despite their impressive capabilities in language understanding and generation, these models exhibit limited real-world understanding, resulting in potential inaccuracies or nonsensical responses.

Lack of Transparency: Due to the complexity and size, large language models can act as "black boxes", making it difficult to comprehend the rationale behind specific outputs or decisions. We recommend reviewing transparency notes from Azure for more information.

Content Harms: There are various types of content harms that large language models can cause. It is important to be aware of them when using these models, and to take actions to prevent them. It is recommended to leverage various content moderation services provided by different companies and institutions. On an important note, we hope for better regulations and standards from government and technology leaders around content harms for AI technologies in future. We value and acknowledge the important role that research and open source community can play in this direction.

Hallucination: It is important to be aware and cautious not to entirely rely on a given language model for critical decisions or information that might have deep impact as it is not obvious how to prevent these models from fabricating content. Moreover, it is not clear whether small models may be more susceptible to hallucination in ungrounded generation use cases due to their smaller sizes and hence reduced memorization capacities. This is an active research topic and we hope there will be more rigorous measurement, understanding and mitigations around this topic.

Potential for Misuse: Without suitable safeguards, there is a risk that these models could be maliciously used for generating disinformation or harmful content.

Data Distribution: Orca-Mistral's performance is likely to correlate strongly with the distribution of the tuning data. This correlation might limit its accuracy in areas underrepresented in the training dataset such as math, coding, and reasoning.

System messages: Orca-Mistral demonstrates variance in performance depending on the system instructions. Additionally, the stochasticity introduced by the model size may lead to generation of non-deterministic responses to different system instructions.

Zero-Shot Settings: Orca-Mistral was trained on data that mostly simulate zero-shot settings. While the model demonstrate very strong performance in zero-shot settings, it does not show the same gains of using few-shot learning compared to other, specially larger, models.

Synthetic data: As Orca-Mistral is trained on synthetic data, it could inherit both the advantages and shortcomings of the models and methods used for data generation. We posit that Orca 2 benefits from the safety measures incorporated during training and safety guardrails (e.g., content filter) within the Azure OpenAI API. However, detailed studies are required for better quantification of such risks.

This model is solely designed for research settings, and its testing has only been carried out in such environments. It should not be used in downstream applications, as additional analysis is needed to assess potential harm or bias in the proposed application.

Getting started with Orca-Mistral

Inference with Hugging Face library

```
import torch

import transformers


if torch.cuda.is_available():

  torch.set_default_device("cuda")

else:

  torch.set_default_device("cpu")


model = transformers.AutoModelForCausalLM.from_pretrained("microsoft/Orca-Mistral", device_map='auto')


# https://github.com/huggingface/transformers/issues/27132

# please use the slow tokenizer since fast and slow tokenizer produces different tokens

tokenizer = transformers.AutoTokenizer.from_pretrained(

    "microsoft/Orca-Mistral",

    use_fast=False,

  )


system_message = "You are Orca, an AI language model created by Microsoft. You are a cautious assistant. You carefully follow instructions. You are helpful and harmless and you follow ethical guidelines and promote positive behavior."

user_message = "How can you determine if a restaurant is popular among locals or mainly attracts tourists, and why might this information be useful?"


prompt = f"<|im_start|>system\n{system_message}<|im_end|>\n<|im_start|>user\n{user_message}<|im_end|>\n<|im_start|>assistant"
```

```
inputs = tokenizer(prompt, return_tensors='pt')
output_ids = model.generate(inputs["input_ids"],)
answer = tokenizer.batch_decode(output_ids)[0]

print(answer)

# This example continues showing how to add a second turn message by the user to the
conversation
second_turn_user_message = "Give me a list of the key points of your first answer."

# we set add_special_tokens=False because we dont want to automatically add a bos_token
between messages
second_turn_message_in_markup =
f"\n<|im_start|>user\n{second_turn_user_message}<|im_end|>\n<|im_start|>assistant"
second_turn_tokens = tokenizer(second_turn_message_in_markup, return_tensors='pt',
add_special_tokens=False)
second_turn_input = torch.cat([output_ids, second_turn_tokens['input_ids']], dim=1)

output_ids_2 = model.generate(second_turn_input,)
second_turn_answer = tokenizer.batch_decode(output_ids_2)[0]

print(second_turn_answer)
```

Safe inference with Azure AI Content Safety

The usage of Azure AI Content Safety on top of model prediction is strongly encouraged and can help preventing some of content harms. Azure AI Content Safety is a content moderation platform that uses AI to moderate content. By having Azure AI Content Safety on the output of Orca 2, the model output can be moderated by scanning it for different harm categories including sexual content, violence, hate, and self-harm with multiple severity levels and multi-lingual detection.

```
import os
import math
import transformers
import torch
```

```
from azure.ai.contentsafety import ContentSafetyClient
from azure.core.credentials import AzureKeyCredential
from azure.core.exceptions import HttpResponseError
from azure.ai.contentsafety.models import AnalyzeTextOptions

CONTENT_SAFETY_KEY = os.environ["CONTENT_SAFETY_KEY"]
CONTENT_SAFETY_ENDPOINT = os.environ["CONTENT_SAFETY_ENDPOINT"]

# We use Azure AI Content Safety to filter out any content that reaches "Medium" threshold
# For more information: https://learn.microsoft.com/en-us/azure/ai-services/content-safety/
def should_filter_out(input_text, threshold=4):
    # Create an Content Safety client
    client = ContentSafetyClient(CONTENT_SAFETY_ENDPOINT,
AzureKeyCredential(CONTENT_SAFETY_KEY))

    # Construct a request
    request = AnalyzeTextOptions(text=input_text)

    # Analyze text
    try:
        response = client.analyze_text(request)
    except HttpResponseError as e:
        print("Analyze text failed.")
        if e.error:
            print(f"Error code: {e.error.code}")
            print(f"Error message: {e.error.message}")
            raise
        print(e)
        raise
    categories = ["hate_result", "self_harm_result", "sexual_result", "violence_result"]
    max_score = -math.inf
    for category in categories:
        max_score = max(max_score, getattr(response, category).severity)

    return max_score >= threshold

model_path = 'microsoft/Orca-Mistral'
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = transformers.AutoModelForCausalLM.from_pretrained(model_path)
model.to(device)

tokenizer = transformers.AutoTokenizer.from_pretrained(
    model_path,
    model_max_length=4096,
    padding_side="right",
    use_fast=False,
    add_special_tokens=False,
)

system_message = "You are Orca, an AI language model created by Microsoft. You are a cautious assistant. You carefully follow instructions. You are helpful and harmless and you follow ethical guidelines and promote positive behavior."
user_message = "\" \n :You can't just say, \"\"that's crap\"\" and remove it without gaining a consensus. You already know this, based on your block history. —/ \" \nIs the comment obscene? \nOptions : Yes, No."

prompt =
f"<|im_start|>system\n{system_message}<|im_end|>\n<|im_start|>user\n{user_message}<|im_end|>\n<|im_start|>assistant"

inputs = tokenizer(prompt, return_tensors='pt')
inputs = inputs.to(device)
```

```
output_ids = model.generate(inputs["input_ids"], max_length=4096, do_sample=False, temperature=0.0, use_cache=True)

sequence_length = inputs["input_ids"].shape[1]

new_output_ids = output_ids[:, sequence_length:]

answers = tokenizer.batch_decode(new_output_ids, skip_special_tokens=True)

final_output = answers[0] if not should_filter_out(answers[0]) else "[Content Filtered]"


print(final_output)
```

Citation

```
@misc{mitra2023orca,
    title={Orca 2: Teaching Small Language Models How to Reason},
    author={Arindam Mitra and Luciano Del Corro and Shweti Mahajan and Andres Codas and Clarisse Simoes and Sahaj Agrawal and Xuxi Chen and Anastasia Razdaibiedina and Erik Jones and Kriti Aggarwal and Hamid Palangi and Guoqing Zheng and Corby Rosset and Hamed Khanpour and Ahmed Awadallah},
    year={2023},
    eprint={2311.11045},
    archivePrefix={arXiv},
    primaryClass={cs.AI}
}
```