Demystifying LLMs

Devendra Singh Chaplot
Mistral AI

Feb 13, 2024
Mistral AI

Co-Founders

Arthur Mensch
CEO
Former AI researcher at DeepMind, Polytechnique alum

Timothée Lacroix
CTO
Former AI researcher at Meta, ENS alum

Guillaume Lample
Chief Scientist
Former AI Researcher at Meta, Polytechnique alum

Releases

La plateforme
Dec 11, 2023  By Mistral AI team
Our first AI endpoints are available in early access.
Read More

Mixtral of experts
Dec 11, 2023  By Mistral AI team
A high quality Sparse Mixture-of-Experts.
Read More

Mistral 7B
Sep 27, 2023  By Mistral AI team
The best 7B model to date, Apache 2.0
Read More

$500M+ funding, Offices in Paris/London/SF Bay Area
Our models.

**MISTRAL 7B**
A 7B dense Transformer, fast-deployed and easily customisable. Small, yet powerful for a variety of use cases. Supports English and code, and a 8k context window.

Licence: **Apache 2.0**

[Download it](#)

**MIXTRAL 8X7B**
A 7B sparse Mixture-of-Experts model with stronger capabilities than Mistral 7B. Uses 12B active parameters out of 45B total. Supports multiple languages, code and 32k context window.

Licence: **Apache 2.0**

[Download it](#)
Contents

• Stages of LLM Training:
  • Pretraining
  • Instruction-Tuning
  • Learning from Human Preferences: DPO/RLHF
• Evaluation of LLMs
• Retrieval Augmented Generation (RAG)
  • Recipe for RAG with code
Stages of LLM Training

1. Pretraining
2. Instruction-Tuning
3. Learning from Human Feedback
Stages of LLM Training

1. Pretraining
2. Instruction-Tuning
3. Learning from Human Feedback
Pretraining

Mistral of Experts

Abstract

We introduce Mistral 8x7B, a Sparse Mixture of Experts (SMeL) language model. Mistral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e., experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 4TB parameters, but only uses 12B active parameters during inference. Mistral was trained with a context size of 32K tokens and it outperformed or matches Llama 2 70B and GPT3.5 across all evaluated benchmarks. In particular, Mistral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mistral 8x7B – Instinct, that surpasses GPT3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.

Code: https://github.com/mistralai/mistral-src
We introduce Mixtral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mixtral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e., experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mixtral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mixtral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mixtral 8x7B – Instruct, that surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.
We introduce Mixtral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mixtral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e. experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mixtral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mixtral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mixtral 8x7B – Instruct, that surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.
We introduce Mixtral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mixtral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e. experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mixtral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mixtral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mixtral 8x7B – Instruct, that surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.
We introduce Mixtral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mixtral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e. experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mixtral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mixtral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mixtral 8x7B – Instruct, that surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.
We introduce Mixtral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mixtral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e. experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mixtral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mixtral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mixtral 8x7B – Instruct, that surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.

Abstract

We introduce Mistral 8x7B, a Sparse Mixture of Experts (SMoE) language model. Mistral has the same architecture as Mistral 7B, with the difference that each layer is composed of 8 feedforward blocks (i.e. experts). For every token, at each layer, a router network selects two experts to process the current state and combine their outputs. Even though each token only sees two experts, the selected experts can be different at each timestep. As a result, each token has access to 47B parameters, but only uses 13B active parameters during inference. Mistral was trained with a context size of 32k tokens and it outperforms or matches Llama 2 70B and GPT-3.5 across all evaluated benchmarks. In particular, Mistral vastly outperforms Llama 2 70B on mathematics, code generation, and multilingual benchmarks. We also provide a model finetuned to follow instructions, Mistral 8x7B – Instruct, that surpasses GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B – chat model on human benchmarks. Both the base and instruct models are released under the Apache 2.0 license.

Code: https://github.com/mistralai/mistral-src
Pretraining

- Task: Next token prediction
- 1 token ~= 0.75 word
- Vocab size: O(10K) tokens
- Each token is represented by an integer

We introduce Mixtral 8x7B

Large Language Model (LLM)

We introduce Mixtral
Pretraining

- Why is it hard?
Pretraining

• Why is it hard?
  • **Time:** Datasets are huge - $O(1T)$ tokens
  • Preprocessing, Cleaning, Deduplication
  • More data might not lead to better model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sampling prop.</th>
<th>Epochs</th>
<th>Disk size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommonCrawl</td>
<td>67.0%</td>
<td>1.10</td>
<td>3.3 TB</td>
</tr>
<tr>
<td>C4</td>
<td>15.0%</td>
<td>1.06</td>
<td>783 GB</td>
</tr>
<tr>
<td>Github</td>
<td>4.5%</td>
<td>0.64</td>
<td>328 GB</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>4.5%</td>
<td>2.45</td>
<td>83 GB</td>
</tr>
<tr>
<td>Books</td>
<td>4.5%</td>
<td>2.23</td>
<td>85 GB</td>
</tr>
<tr>
<td>ArXiv</td>
<td>2.5%</td>
<td>1.06</td>
<td>92 GB</td>
</tr>
<tr>
<td>StackExchange</td>
<td>2.0%</td>
<td>1.03</td>
<td>78 GB</td>
</tr>
</tbody>
</table>

Llama pretraining data mixture
Pretraining

- **Why is it hard?**
  - **Time:** Datasets are huge - $O(1T)$ tokens
    - Preprocessing, Cleaning, Deduplication
  - **Money:** $O(1-100B)$ parameters
    - $O(1-10K)$ GPUs for weeks or months
    - $O(10-100M)$ per model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sampling prop.</th>
<th>Epochs</th>
<th>Disk size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommonCrawl</td>
<td>67.0%</td>
<td>1.10</td>
<td>3.3 TB</td>
</tr>
<tr>
<td>C4</td>
<td>15.0%</td>
<td>1.06</td>
<td>783 GB</td>
</tr>
<tr>
<td>Github</td>
<td>4.5%</td>
<td>0.64</td>
<td>328 GB</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>4.5%</td>
<td>2.45</td>
<td>83 GB</td>
</tr>
<tr>
<td>Books</td>
<td>4.5%</td>
<td>2.23</td>
<td>85 GB</td>
</tr>
<tr>
<td>ArXiv</td>
<td>2.5%</td>
<td>1.06</td>
<td>92 GB</td>
</tr>
<tr>
<td>StackExchange</td>
<td>2.0%</td>
<td>1.03</td>
<td>78 GB</td>
</tr>
</tbody>
</table>

Llama pretraining data mixture
Pretraining

• Why is it hard?
  • **Time**: Datasets are huge - $O(1T)$ tokens
    • Preprocessing, Cleaning, Deduplication
  • More data might not lead to better model
  • **Money**: $O(1-100B)$ parameters
    • $O(1-10K)$ GPUs for weeks or months
    • $O(10-100M)$ per model
  • **YOLO**: decide model architecture, hyperparameters, data mixture for the “big run”
    • Best hyper-parameters for a smaller model might not be the best for a larger model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sampling prop.</th>
<th>Epochs</th>
<th>Disk size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommonCrawl</td>
<td>67.0%</td>
<td>1.10</td>
<td>3.3 TB</td>
</tr>
<tr>
<td>C4</td>
<td>15.0%</td>
<td>1.06</td>
<td>783 GB</td>
</tr>
<tr>
<td>Github</td>
<td>4.5%</td>
<td>0.64</td>
<td>328 GB</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>4.5%</td>
<td>2.45</td>
<td>83 GB</td>
</tr>
<tr>
<td>Books</td>
<td>4.5%</td>
<td>2.23</td>
<td>85 GB</td>
</tr>
<tr>
<td>ArXiv</td>
<td>2.5%</td>
<td>1.06</td>
<td>92 GB</td>
</tr>
<tr>
<td>StackExchange</td>
<td>2.0%</td>
<td>1.03</td>
<td>78 GB</td>
</tr>
</tbody>
</table>

**Llama pretraining data mixture**

<table>
<thead>
<tr>
<th>params</th>
<th>dimension</th>
<th>n heads</th>
<th>n layers</th>
<th>learning rate</th>
<th>batch size</th>
<th>n tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.7B</td>
<td>4096</td>
<td>32</td>
<td>32</td>
<td>$3.0e^{-4}$</td>
<td>4M</td>
<td>1.0T</td>
</tr>
<tr>
<td>13.0B</td>
<td>5120</td>
<td>40</td>
<td>40</td>
<td>$3.0e^{-4}$</td>
<td>4M</td>
<td>1.0T</td>
</tr>
<tr>
<td>32.5B</td>
<td>6656</td>
<td>52</td>
<td>60</td>
<td>$1.5e^{-4}$</td>
<td>4M</td>
<td>1.4T</td>
</tr>
<tr>
<td>65.2B</td>
<td>8192</td>
<td>64</td>
<td>80</td>
<td>$1.5e^{-4}$</td>
<td>4M</td>
<td>1.4T</td>
</tr>
</tbody>
</table>

**Llama Model sizes, architectures, and optimization hyper-parameters.**
Can we use the Pretrained model?
Prompt:

Write a python function to find whether the input number is prime.
Prompt:
Write a python function to find whether the input number is prime.

Response:
Example:
Input: 17
Output: True

Input: 15
Output: False

Approach:
1. Let’s assume that the input number is n
2. Check if n is divisible by 1. If n is divisible by 1 then it is not a prime number.
3. Check if n is divisible by 2. If n is divisible by 2 then it is not a prime number.
Can we use the Pretrained model?

Prompt:

```python
def is_prime(x: int):
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
```
def is_prime(x: int):
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
Can we use the Pretrained model?

Prompt:

```python
def is_prime(x: int):
    
    takes as input an integer x. Returns True if x is prime and False otherwise
```

Response:

```python
if x <= 1:
    return False
for i in range(2, int(x ** 0.5) + 1):
    if x % i == 0:
        return False
return True
```

Model knows the answer but it is not aligned with human preferences
Stages of LLM Training

1. Pretraining
2. Instruction-Tuning
3. Learning from Human Feedback
### Prompt:

Write a python function to find whether the input number is prime. 

```python
def is_prime(x: int):
    
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

### Response:
Prompt:

[INST] Write a python function to find whether the input number is prime. [INST]

Response:

```python
def is_prime(x: int):
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```
Write a python function to find whether the input number is prime.

```python
def is_prime(x: int):
    
    takes as input an integer x. Returns True if x is prime and False otherwise
    
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

Large Language Model

O(1-100B) parameters
**Prompt:**

Write a python function to find whether the input number is prime.

```python
def is_prime(x: int):
    
    # Takes as input an integer x. Returns True if x is prime and False otherwise
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

**Response:**

A Python function `is_prime` is defined to check if a given number `x` is prime. The function takes an integer `x` as input and returns `True` if `x` is prime and `False` otherwise. The function checks if `x` is less than or equal to 1 and returns `False` in that case. If `x` is greater than 1, it checks for divisors from 2 to the square root of `x` (inclusive). If a divisor is found, it returns `False`. If no divisor is found, it returns `True`. The function uses a for loop to iterate through possible divisors and checks if `x` is divisible by any of them.
def is_prime(x: int):
    """    takes as input an integer x. Returns True if x is prime and False otherwise    """
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
Instruction Fine-tuning

- **Dataset**
  - Paired: (Prompt, Response)
  - $O(10-100K)$ instructions
Instruction Fine-tuning

- **Dataset**
  - Paired: (Prompt, Response)
  - O(10-100K instructions)

- **Task:**
  - Next word prediction (Masked)
Instruction Fine-tuning

- **Dataset**
  - Paired: (Prompt, Response)
  - $O(10$-$100K$ instructions)
- **Task**:
  - Next word prediction (Masked)
- **Compute**:
  - $O(1$-$100$) GPUs
  - Few hrs/days
Stages of LLM Training

1. Pretraining
2. Instruction-Tuning
3. Learning from Human Feedback
Human Preferences

Human preferences are cheaper/easier than human annotation

Prompt:

```python
def is_prime(x: int):
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

Response 1:

```python
def is_prime(x: int):
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

Response 2:

```python
def is_prime(x: int):
    """
    takes as input an integer x. Returns True if x is prime and False otherwise
    """
    if x <= 1:
        return False
    for i in range(2, x):
        if x % i == 0:
            return False
    return True
```
Human Preferences

Human preferences are cheaper/easier than human annotation

Prompt:

[INST] Write a python function to find whether the input number is prime. [INST]

```python
def is_prime(x: int):
    
    takes as input an integer x. Returns True if x is prime and False otherwise
    
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

Response 1:

```python
def is_prime(x: int):
    
    takes as input an integer x. Returns True if x is prime and False otherwise
    
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

Response 2:

```python
def is_prime(x: int):
    
    takes as input an integer x. Returns True if x is prime and False otherwise
    
    if x <= 1:
        return False
    for i in range(2, x):
        if x % i == 0:
            return False
    return True
```

Response 1 > Response 2
Reinforcement Learning from Human Feedback (RLHF)

[Deep Reinforcement Learning from Human Preferences. Christiano et al. 2017]
Direct Preference Optimization (DPO)

Reinforcement Learning from Human Feedback (RLHF)
- Preference data
- "write me a poem about the history of jazz"
- Maximum likelihood
- Label rewards
- Reward model
- Sample completions
- LM policy
- Reinforcement learning

Direct Preference Optimization (DPO)
- Preference data
- "write me a poem about the history of jazz"
- Maximum likelihood
- Final LM

[Deep Reinforcement Learning from Human Preferences. Christiano et al. 2017]
[Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Rafailov et al. 2023]
Stages of LLM Training

**Pretraining**
- **Dataset:**
  - Raw text
  - Few trillions of tokens
- **Task:**
  - Next word prediction
- **Compute:**
  - $O(1-10K)$ GPUs
  - Weeks/months of training

**Instruction-Tuning**
- **Dataset:**
  - Paired: (Prompt, Response)
  - $O(10-100K)$ instructions
- **Task:**
  - Next word prediction (Masked)
- **Compute:**
  - $O(1-100)$ GPUs
  - Few hrs/days

**Learning from Human Feedback**
- **Dataset:**
  - Human Preference Data
  - $O(10-100K)$
- **Task:**
  - RLHF/DPO
- **Compute:**
  - $O(1-100)$ GPUs
  - Few hrs/days
Evaluation of LLMs
Evaluation of pretrained models

- **Commonsense Reasoning (0-shot):** Hellaswag [32], Winogrande [26], PIQA [3], SIQA [27], OpenbookQA [22], ARC-Easy, ARC-Challenge [8], CommonsenseQA [30]
- **World Knowledge (5-shot):** NaturalQuestions [20], TriviaQA [19]
- **Reading Comprehension (0-shot):** BoolQ [7], QuAC [5]
- **Math:** GSM8K [9] (8-shot) with maj@8 and MATH [17] (4-shot) with maj@4
- **Code:** Humaneval [4] (0-shot) and MBPP [1] (3-shot)
- **Popular aggregated results:** MMLU [16] (5-shot), BBH [29] (3-shot), and AGI Eval [34] (3-5-shot, English multiple-choice questions only)
Evaluation of pretrained models

- **Commonsense Reasoning (0-shot)**: Hellaswag [32], Winogrande [26], PIQA [3], SIQA [27], OpenbookQA [22], ARC-Easy, ARC-Challenge [8], CommonsenseQA [30]
- **World Knowledge (5-shot)**: NaturalQuestions [20], TriviaQA [19]
- **Reading Comprehension (0-shot)**: BoolQ [7], QuAC [5]
- **Math**: GSM8K [9] (8-shot) with maj@8 and MATH [17] (4-shot) with maj@4
- **Code**: Humaneval [4] (0-shot) and MBPP [1] (3-shot)
- **Popular aggregated results**: MMLU [16] (5-shot), BBH [29] (3-shot), and AGI Eval [34] (3-5-shot, English multiple-choice questions only)

0-shot:

```python
def is_prime(x: int):
    ***
    takes as input an integer x.
    Returns True if x is prime and False otherwise
    ***
```
Evaluation of pretrained models

- **Commonsense Reasoning (0-shot):** Hellaswag [32], Winogrande [26], PIQA [3], SIQA [27], OpenbookQA [22], ARC-Easy, ARC-Challenge [8], CommonsenseQA [30]
- **World Knowledge (5-shot):** NaturalQuestions [20], TriviaQA [19]
- **Reading Comprehension (0-shot):** BoolQ [7], QuAC [5]
- **Math:** GSM8K [9] (8-shot) with maj@8 and MATH [17] (4-shot) with maj@4
- **Code:** Humaneval [4] (0-shot) and MBPP [1] (3-shot)
- **Popular aggregated results:** MMLU [16] (5-shot), BBH [29] (3-shot), and AGI Eval [34] (3-5-shot, English multiple-choice questions only)

### 0-shot:

```python
def is_prime(x: int):
    ***
    takes as input an integer x.
    Returns True if x is prime and False otherwise
    ***
```

### 3-shot:

```python
## How old is Barack Obama in 2014?
Barack Obama is 57 years old in 2014.

## What is Barack Obama’s birthday?
Barack Obama was born on August 4, 1961.

## What is the name of Barack Obama’s wife?
Barack Obama’s wife is Michelle Obama.

## How tall is Barack Obama?
```
Evaluation of pretrained models

- **Commonsense Reasoning (0-shot):** Hellaswag [32], Winogrande [26], PIQA [3], SIQA [27], OpenbookQA [22], ARC-Easy, ARC-Challenge [8], CommonsenseQA [30]
- **World Knowledge (5-shot):** NaturalQuestions [20], TriviaQA [19]
- **Reading Comprehension (0-shot):** BoolQ [7], QuAC [5]
- **Math:** GSM8K [9] (8-shot) with maj@8 and MATH [17] (4-shot) with maj@4
- **Code:** Humaneval [4] (0-shot) and MBPP [1] (3-shot)
- **Popular aggregated results:** MMLU [16] (5-shot), BBH [29] (3-shot), and AGI Eval [34] (3-5-shot, English multiple-choice questions only)

0-shot:

```python
def is_prime(x: int):
    ***
    takes as input an integer x.
    Returns True if x is prime and False otherwise
    ***
```

3-shot:

```
## How old is Barack Obama in 2014?
Barack Obama is 57 years old in 2014.

## What is Barack Obama’s birthday?
Barack Obama was born on August 4, 1961.

## What is the name of Barack Obama’s wife?
Barack Obama’s wife is Michelle Obama.

## How tall is Barack Obama?
```
Evaluation of Instruction-tuned models

Write a python function to find whether the input number is prime.

Here is a Python function to check if a given number is prime:

```python
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True
```

A number is considered prime if it has exactly two factors: 1 and itself. All numbers greater than 1 have at least three factors, one of which is 1, so we can start our range from 2.

Here is a Python function that checks whether a number is prime:

```python
def is_prime(number):
    # We start from 2 because 1 and the number itself are always factors.
    for i in range(2, int(number ** 0.5) + 1):
        # If we find a factor, the number is not prime
        if number % i == 0:
            return False
    # If no factors are found, the number is prime
    return True
```

This function takes an integer as input and returns a boolean value. It starts by checking if the number is less than or equal to 1. If so, it immediately returns `True` for 2 and `False` for all others because these are the only prime numbers below 12.

If the number is greater than 1, the function then checks for factors from 2 up to the square root.
### Evaluation of Instruction-tuned models

<table>
<thead>
<tr>
<th>Rank</th>
<th>🌟 Model</th>
<th>🌟 Arena Elo</th>
<th>95% CI</th>
<th>🗳️ Votes</th>
<th>🧑‍💻 Organization</th>
<th>📋 License</th>
<th>🕒 Knowledge Cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GPT-4:0125-preview</td>
<td>1253</td>
<td>+10/-11</td>
<td>3922</td>
<td>OpenAI</td>
<td>Proprietary</td>
<td>2023/4</td>
</tr>
<tr>
<td>2</td>
<td>GPT-4:1106-preview</td>
<td>1252</td>
<td>+5/-6</td>
<td>35385</td>
<td>OpenAI</td>
<td>Proprietary</td>
<td>2023/4</td>
</tr>
<tr>
<td>3</td>
<td>Bard (Gemini Pro)</td>
<td>1224</td>
<td>+9/-9</td>
<td>9081</td>
<td>Google</td>
<td>Proprietary</td>
<td>Online</td>
</tr>
<tr>
<td>4</td>
<td>GPT-4:0314</td>
<td>1190</td>
<td>+5/-6</td>
<td>18945</td>
<td>OpenAI</td>
<td>Proprietary</td>
<td>2021/9</td>
</tr>
<tr>
<td>5</td>
<td>GPT-4:0613</td>
<td>1162</td>
<td>+4/-5</td>
<td>29950</td>
<td>OpenAI</td>
<td>Proprietary</td>
<td>2021/9</td>
</tr>
<tr>
<td>6</td>
<td>Mistral Medium</td>
<td>1150</td>
<td>+6/-7</td>
<td>15447</td>
<td>Mistral</td>
<td>Proprietary</td>
<td>Unknown</td>
</tr>
<tr>
<td>7</td>
<td>Claude-1</td>
<td>1149</td>
<td>+6/-6</td>
<td>18189</td>
<td>Anthropic</td>
<td>Proprietary</td>
<td>Unknown</td>
</tr>
<tr>
<td>8</td>
<td>Claude-2.0</td>
<td>1132</td>
<td>+6/-7</td>
<td>12131</td>
<td>Anthropic</td>
<td>Proprietary</td>
<td>Unknown</td>
</tr>
<tr>
<td>9</td>
<td>Gemini_Pro_(Dev_API)</td>
<td>1120</td>
<td>+7/-7</td>
<td>7616</td>
<td>Google</td>
<td>Proprietary</td>
<td>2023/4</td>
</tr>
<tr>
<td>10</td>
<td>Claude-2.1</td>
<td>1119</td>
<td>+5/-6</td>
<td>25494</td>
<td>Anthropic</td>
<td>Proprietary</td>
<td>Unknown</td>
</tr>
<tr>
<td>11</td>
<td>GPT-3.5-Turbo-0613</td>
<td>1118</td>
<td>+5/-5</td>
<td>33617</td>
<td>OpenAI</td>
<td>Proprietary</td>
<td>2021/9</td>
</tr>
<tr>
<td>12</td>
<td>Mistral-8x7b-Instruct-v0.1</td>
<td>1118</td>
<td>+5/-7</td>
<td>15705</td>
<td>Mistral</td>
<td>Apache 2.0</td>
<td>2023/12</td>
</tr>
<tr>
<td>13</td>
<td>Yi-34B-Chat</td>
<td>1115</td>
<td>+7/-8</td>
<td>6718</td>
<td>01 AI</td>
<td>Yi License</td>
<td>2023/6</td>
</tr>
<tr>
<td>14</td>
<td>Gemini_Pro</td>
<td>1114</td>
<td>+7/-8</td>
<td>6969</td>
<td>Google</td>
<td>Proprietary</td>
<td>2023/4</td>
</tr>
<tr>
<td>15</td>
<td>Claude-Instant-1</td>
<td>1109</td>
<td>+4/-7</td>
<td>18689</td>
<td>Anthropic</td>
<td>Proprietary</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

**LMSYS Chatbot Arena Leaderboard**

[https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard](https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard)
Evaluation of Instruction-tuned models

- Proxies for human evaluation:
  - **MT Bench:**
    - Ask GPT-4 to score responses
    - 0.90 correlation with human preferences
  - **Alpaca Eval:**
    - Compare win-rate against GPT-4 (v2)
    - 0.84 correlation with human preferences

![Mega LLM Benchmark X-Correlations @gblazex](chart.png)
Practical tips

- Proprietary vs Open-Source

For proprietary models:
- Prompt Engineering: Few-shot prompting, Chain-of-thought
- Retrieval Augmented Generation (RAG)
Practical tips

- Proprietary vs Open-Source

- For proprietary models:
  - Prompt Engineering: Few-shot prompting, Chain-of-thought
  - Retrieval Augmented Generation (RAG)

- For open-source
  - Everything above
  - Task-specific fine-tuning and DPO: Need data and a bit of compute
Practical tips

- Proprietary vs Open-Source

- For proprietary models:
  - Prompt Engineering: Few-shot prompting, Chain-of-thought
  - Retrieval Augmented Generation (RAG)

- For open-source
  - Everything above
  - Task-specific fine-tuning and DPO: Need data and bit of compute

- Balance performance vs cost (training and inference)
  - Proprietary models higher general-purpose performance
  - Open-source models can beat proprietary models on specific tasks with fine-tuning
  - Proprietary models typically have higher inference cost

<table>
<thead>
<tr>
<th>Task</th>
<th>Open-source</th>
<th>Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mistral-tiny</td>
<td>63.0%</td>
<td>75.3%</td>
</tr>
<tr>
<td>Mistral-small</td>
<td>70.6%</td>
<td>88.0%</td>
</tr>
<tr>
<td>MMLU (MCQ in 57 subjects)</td>
<td>83.1%</td>
<td>88.0%</td>
</tr>
<tr>
<td>HellaSwag (10-shot)</td>
<td>86.7%</td>
<td>89.9%</td>
</tr>
<tr>
<td>ARC Challenge (25-shot)</td>
<td>78.1%</td>
<td>88.0%</td>
</tr>
<tr>
<td>WinoGrande (5-shot)</td>
<td>81.2%</td>
<td>88.0%</td>
</tr>
<tr>
<td>MBPP (pass@1)</td>
<td>30.5%</td>
<td>62.3%</td>
</tr>
<tr>
<td>GSM-8K (5-shot)</td>
<td>36.5%</td>
<td>66.7%</td>
</tr>
<tr>
<td>MT Bench (for instruct models)</td>
<td>7.61</td>
<td>8.61</td>
</tr>
</tbody>
</table>
| Price (per M tokens)        | 0.42€       | 1.8€        | 7.5€
Retrieval Augmented Generation (RAG)
When do we need Retrieval Augmented Generation (RAG)?

- LLM doesn’t know everything, sometimes require task-specific knowledge
- Sometimes you want LLMs to answer queries based on some data source to reduce hallucinations
- Knowledge resource doesn’t fit in the context window of the LLM

[Figure from https://lemaoliu.github.io/retrieval-generation-tutorial/]
Recipe for RAG

Data Preparation

A

Raw Data Sources

B

Information Extraction
(OCR, PDF data extraction, web crawlers, etc.)

C

Chunking

D

Embedding

Vector Database

Embedding

Query

1

Retrieval Augmented Generation

2

Embedding

3

Relevant Data

4

LLM(s)

5

Response

[Figure from https://gradientflow.substack.com/p/best-practices-in-retrieval-augmented]
Basic RAG

Retrieval-augmented generation (RAG) is an AI framework that synergizes the capabilities of LLMs and information retrieval systems. It’s useful to answer questions or generate content leveraging external knowledge. There are two main steps in RAG: 1) retrieval: retrieve relevant information from a knowledge base with text embeddings stored in a vector store; 2) generation: insert the relevant information to the prompt for the LLM to generate information. In this guide, we will walk through a very basic example of RAG with two implementations:

- RAG from scratch with Mistral
- RAG with Mistral and LangChain
- RAG with Mistral and Llamaindex

RAG from scratch

This section aims to guide you through the process of building a basic RAG from scratch. We have two goals: firstly, to offer users a comprehensive understanding of the internal workings of RAG and demystify the underlying mechanisms; secondly, to empower you with the essential foundations needed to build an RAG using the minimum required dependencies.