ChartMoE: Mixture of Diversely Aligned Expert Connector for Chart Understanding

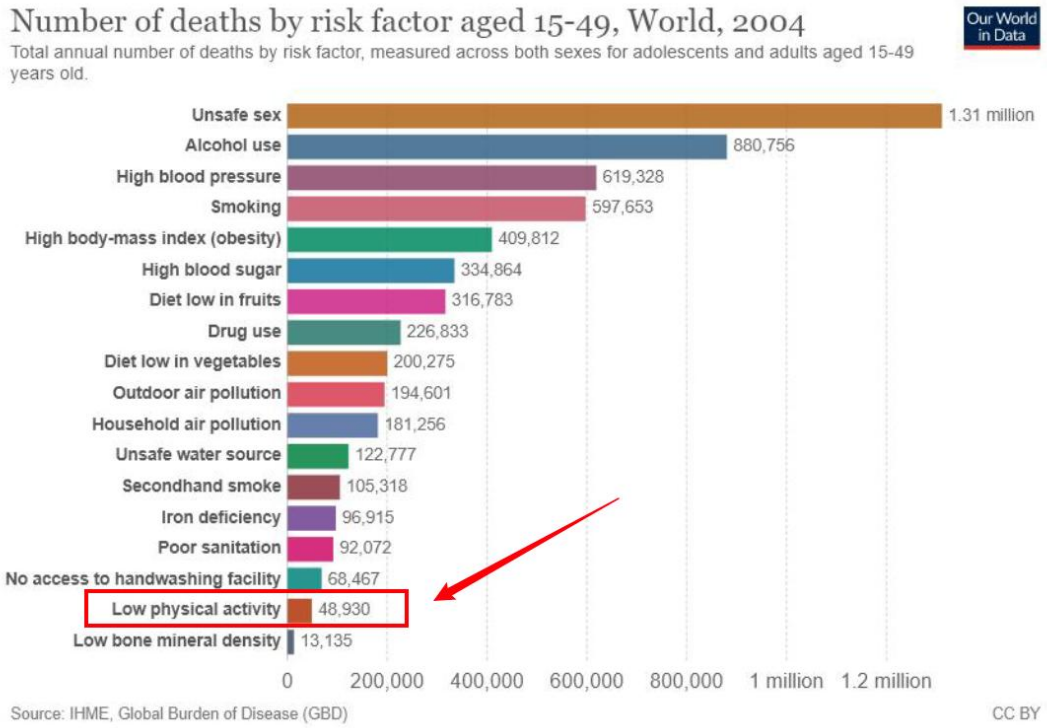
Zhengzhuo Xu, Bowen Qu, Yiyan Qi, Sinan Du, Chengjin Xu,
Chun Yuan, Jian Guo

Presenter : Zhengzhuo Xu

Affiliation : Tsinghua University

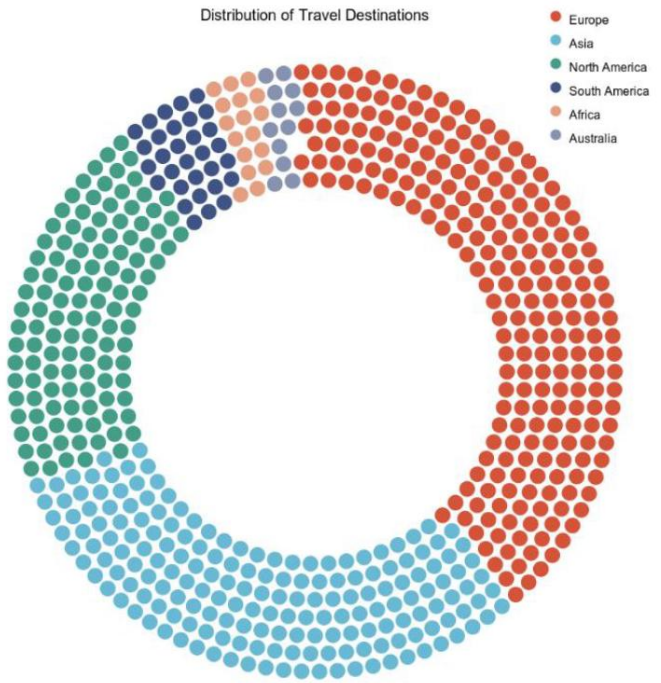
Chart-QA based on MLLM

How many people die because of low physical activity?



ChartQA: OCR task

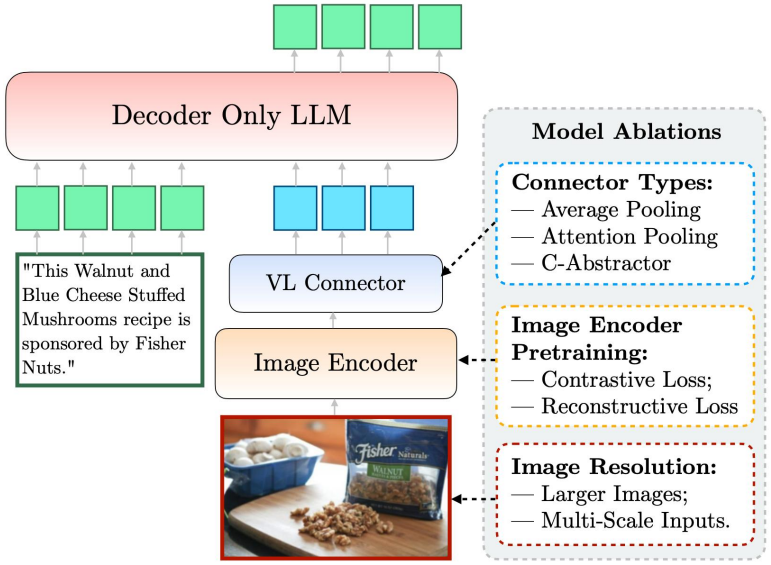
What is the percentageof Asia?



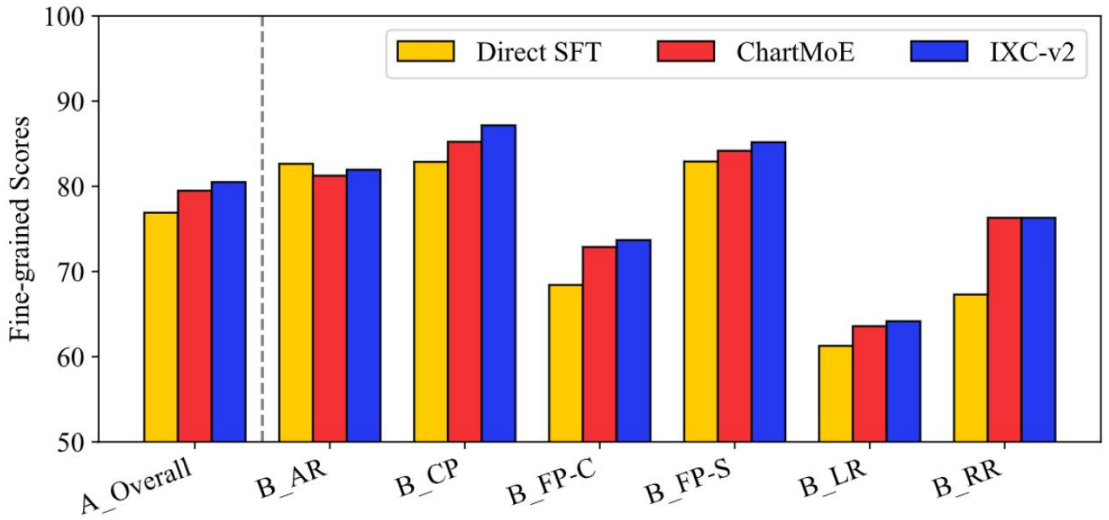
ChartBench: visual reasoning task

Motivation

- Downstream paradigm: aligning the pre-trained MLP projector first
- Directly SFT the linear projector: general capabilities may loss



standard architecture of MLLM (source from MM1[1])

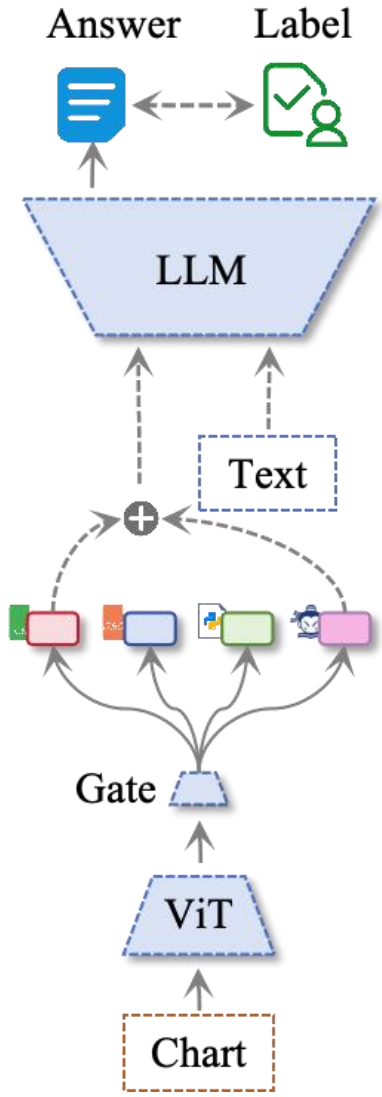


performance on the MMBench

[1]*MM1: Methods, Analysis & Insights from Multimodal LLM Pre-training, ECCV 2024.*

Bridge Modality with MoE!

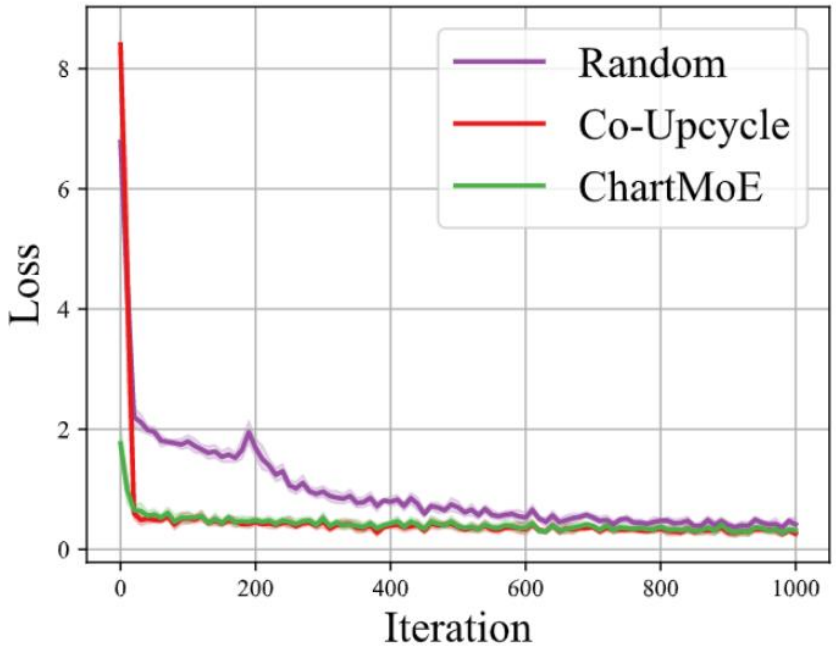
- The connector requires further alignment of proprietary domain data
  - Direct fine-tuning **impairs** general capability and instruct-following
  - How to **sparsely** adapt to downstream tasks?
- **MoE** can work (also a LoRA-like manner!)
  - Keep the original connector as one of the experts
  - Expert **dirversely** initializtion (like lora-bypass)



Our arch

How to Align MoE Connector?

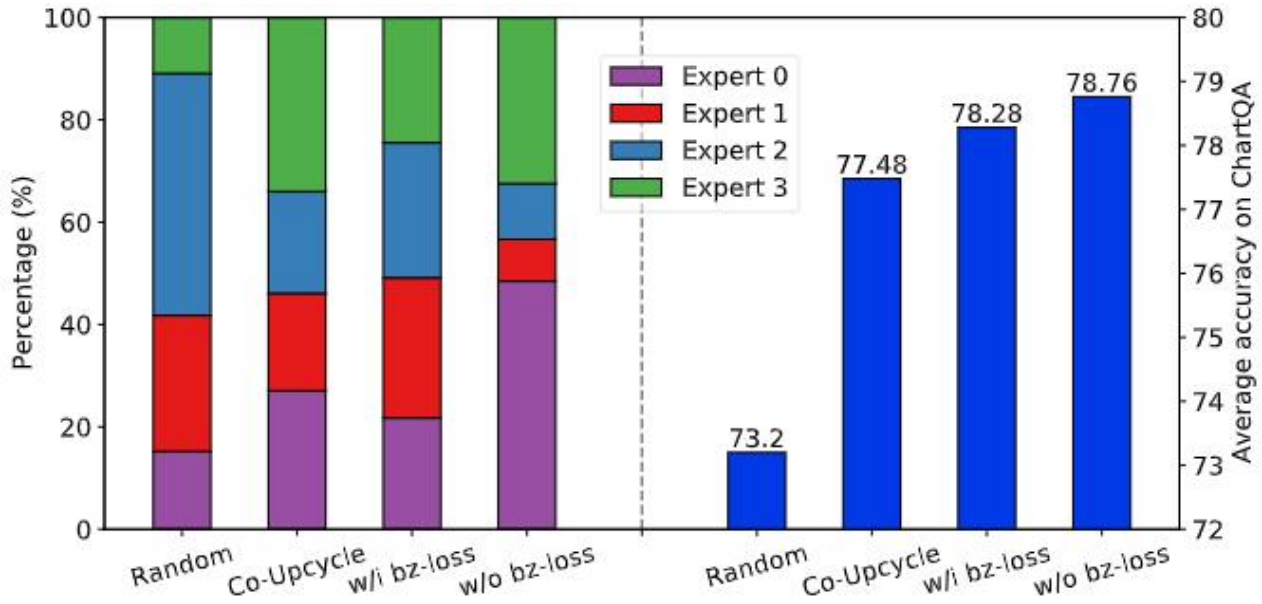
- By **random** initialization?
  - all parameters are randomly initialized
- By **co-upcycle** initialization?
  - copy vanilla parameters to N replicas
- By **diversely** aligned!
  - dirversely initializtion with different alignment



train loss

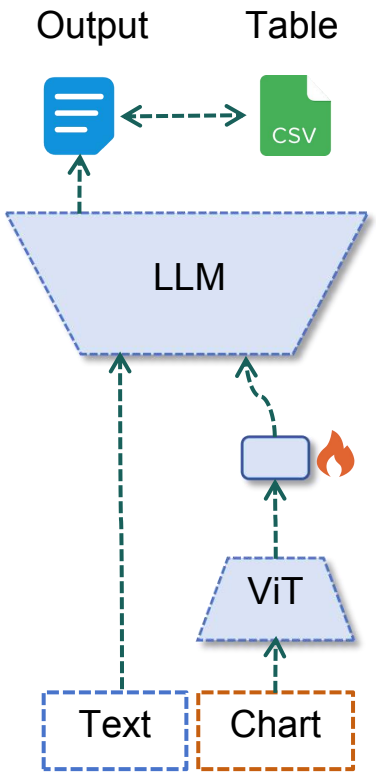
How to Align MoE Connector?

- Why not by **chart types** alignment?
  - load balancing
  - data scarcity
- By **task types** alignment?
  - at what task?
  - paired data?

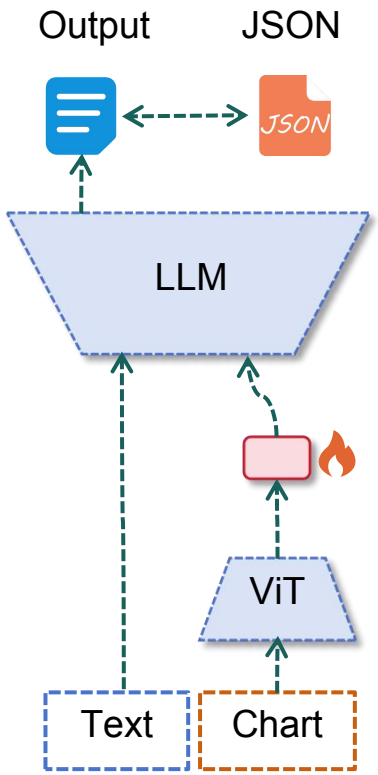


expert distribution and bzloss

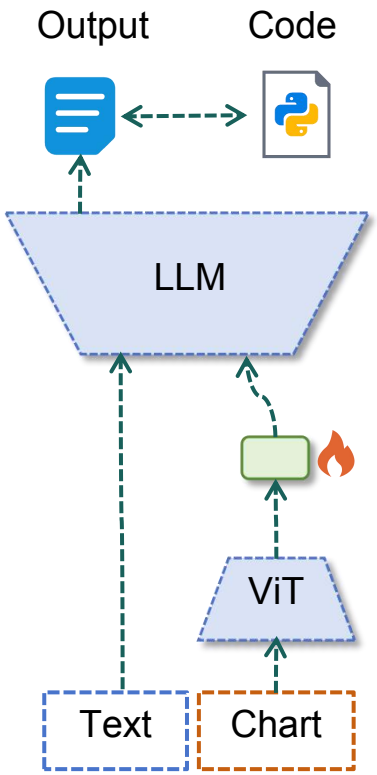
Diversely Alignment



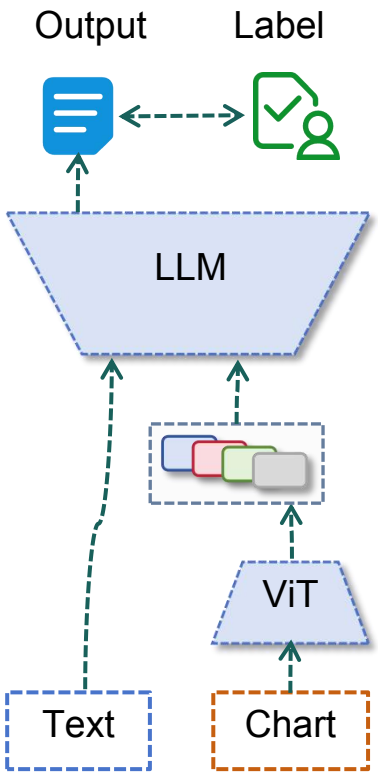
align with **table**



align with **json**

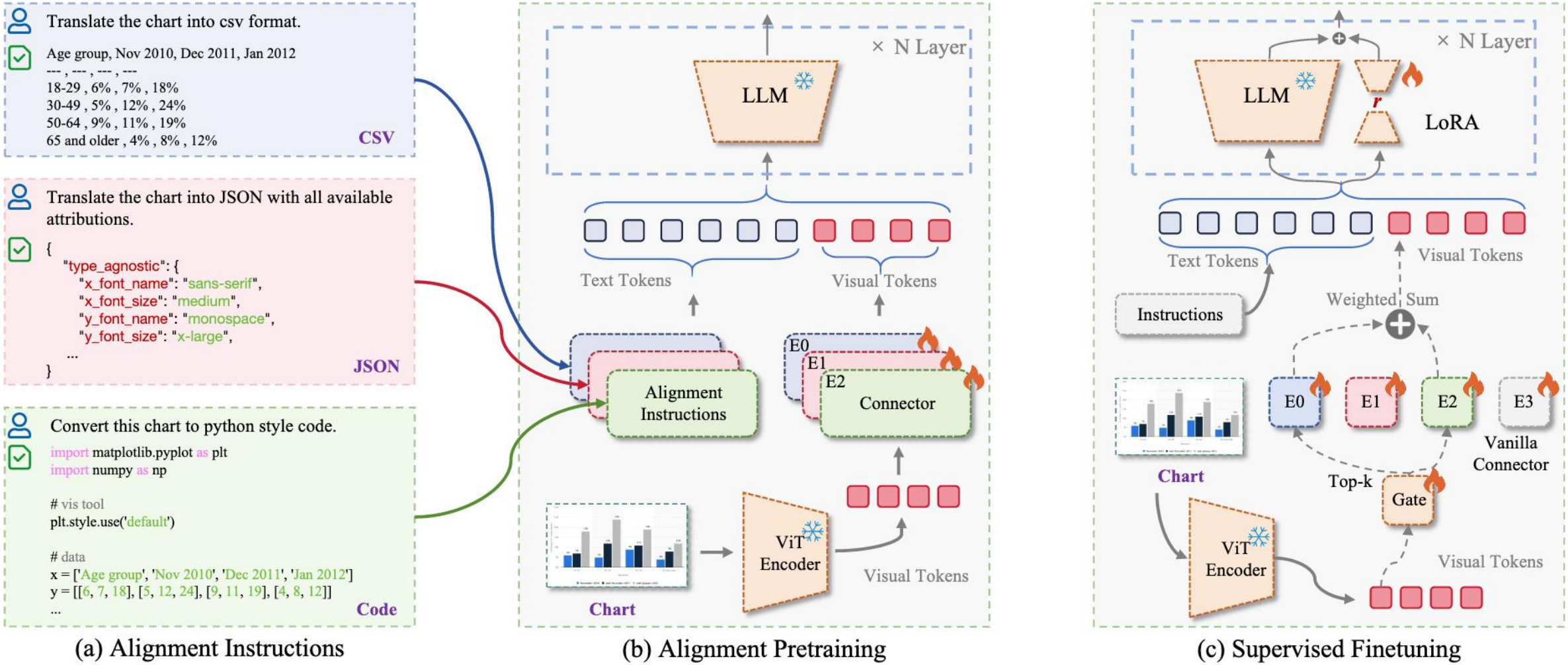


align with **code**



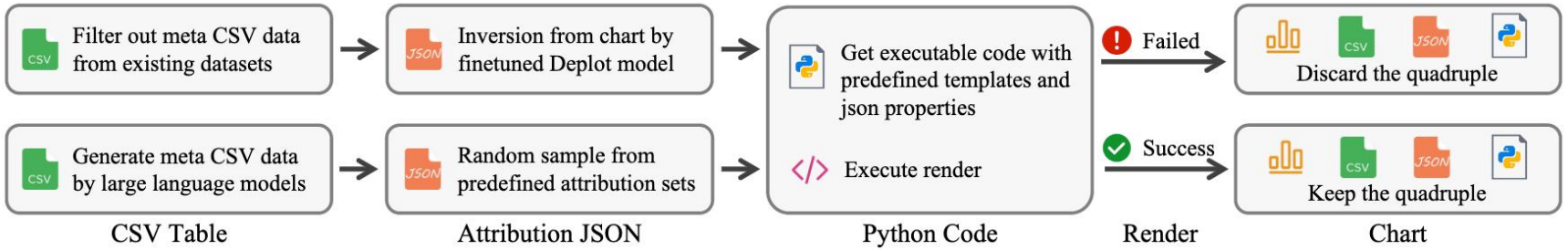
ChartMoE

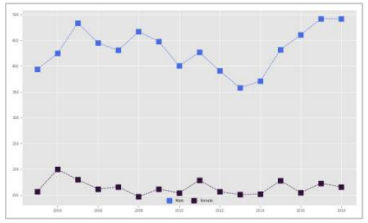
MoE Training Pipeline



(a) Alignment Instructions (b) Alignment Pretraining (c) Supervised Finetuning

Data Collection Pipeline





Chart

```
Characteristic,Male,Female
2019,486,144
2018,492,166
2017,492,173
2016,461,155
2015,432,178
2014,371,152
2013,358,151
2012,391,157
2011,427,179
2010,401,154
2009,448,162
2008,467,147
2007,431,166
2006,445,162
2005,484,180
2004,425,200
2003,394,157
2002,375,207
2001,391,162
2000,397,149
```

CSV

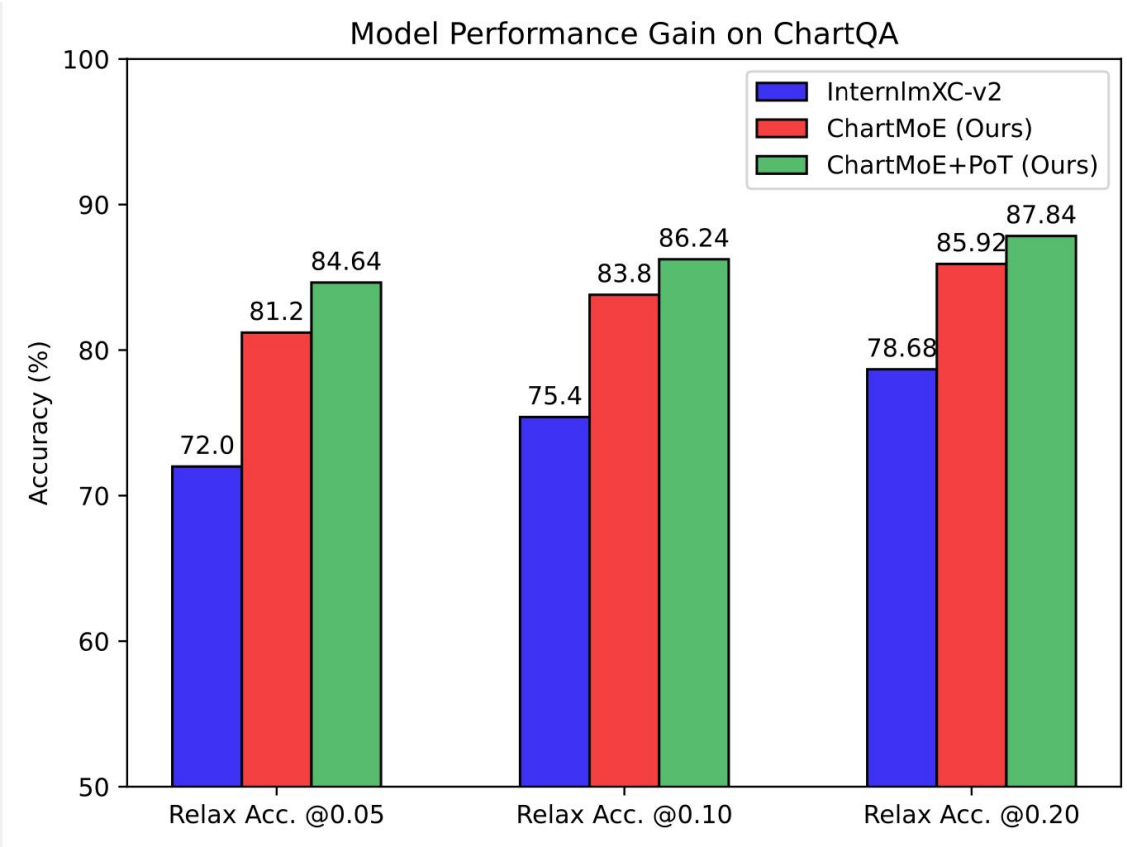
```
{
  "type_agnostic": {
    "x_font_name": "Serif",
    "x_font_size": "large",
    "y_font_name": "sans-serif",
    "y_font_size": "medium",
    "x_tick_size": "x-small",
    "x_tick_rotation": 0,
    "y_tick_size": "large",
    "legend_loc": "lower center",
    "legend_ncols": 2,
    "legend_font_size": "x-small",
    "title_font_name": "monospace",
    "title_font_size": "medium",
    "grid_vis": true,
    "grid_axi": "y",
    "grid_which": "minor",
    "grid_line_style": "solid",
    "vis_tool": "ggplot"
  },
  "type_specific": {
    "colormap": "turbo",
    "marker": "s",
    "style": "--",
    "linewidth": 1.0,
    "markersize": 10
  },
  "layout": {
    "title": "",
    "plot_labels": [
    "Male",
    "Female"
    ]
  }
}
```

JSON

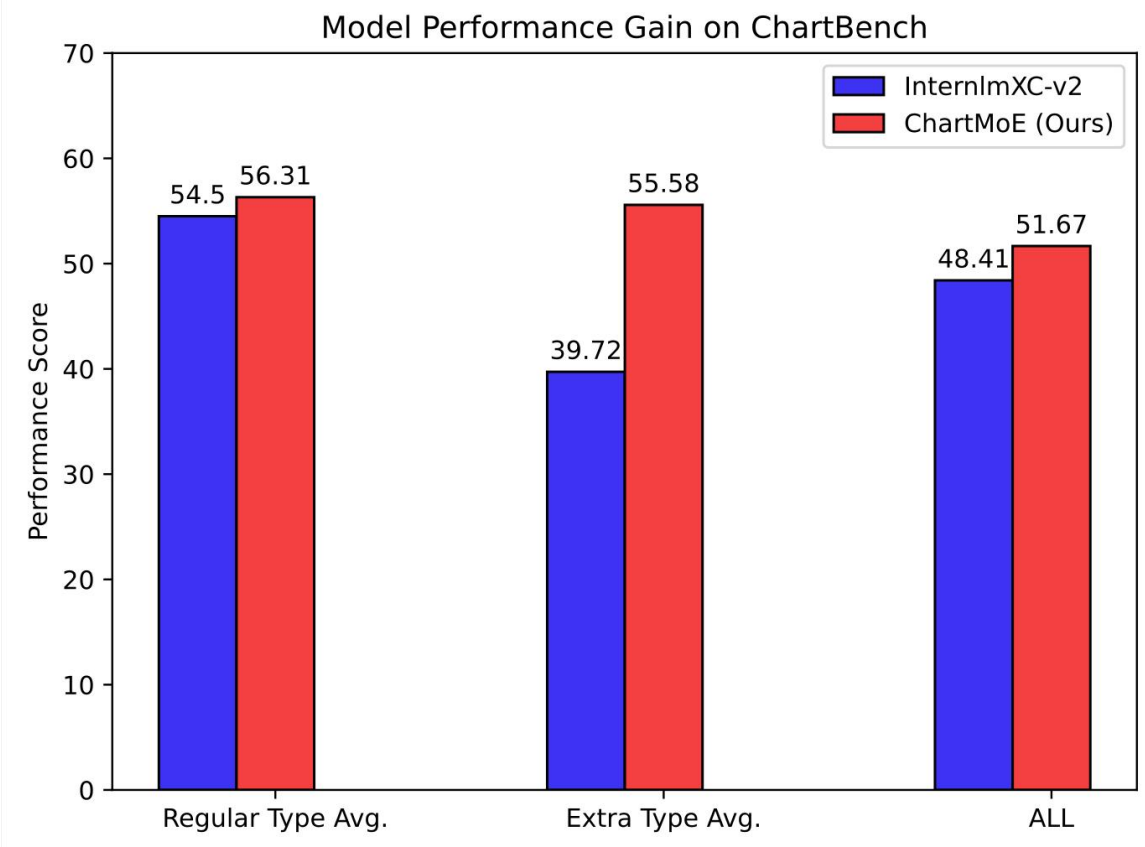
```
import matplotlib.pyplot as plt
import numpy as np

# vis tool
plt.style.use('ggplot')

# data
x = [2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
2016, 2017, 2018]
y = [[394, 425, 484, 445, 431, 467, 448, 401, 427, 391, 358, 371, 432, 461, 492,
492], [157, 200, 180, 162, 166, 147, 162, 154, 179, 157, 151, 152, 178, 155, 173,
166]]

plt.figure(figsize=(10, 6))

# a line chart
plt.plot(x,y[0], label="Male", color='#466be3', marker='s', markersize=10,
linestyle='--', linewidth=1.0)
plt.plot(x,y[1], label="Female", color='#30123b', marker='s', markersize=10,
linestyle='--', linewidth=1.0)

# set the tick of x/y
plt.xticks(fontsize='x-small', rotation=0)
plt.yticks(fontsize='x-small')

# set the global legend
plt.legend(loc='lower center', ncol=2, fontsize='x-small')

# set the grid
plt.grid(visible=True, which='minor', linestyle='solid', axis='y')

# Automatically resize the image by tight_layout()
plt.tight_layout()
# save the chart
plt.savefig('output.png')
# Clear the current image state
plt.clf()
```

Code

Quantitative Analysis



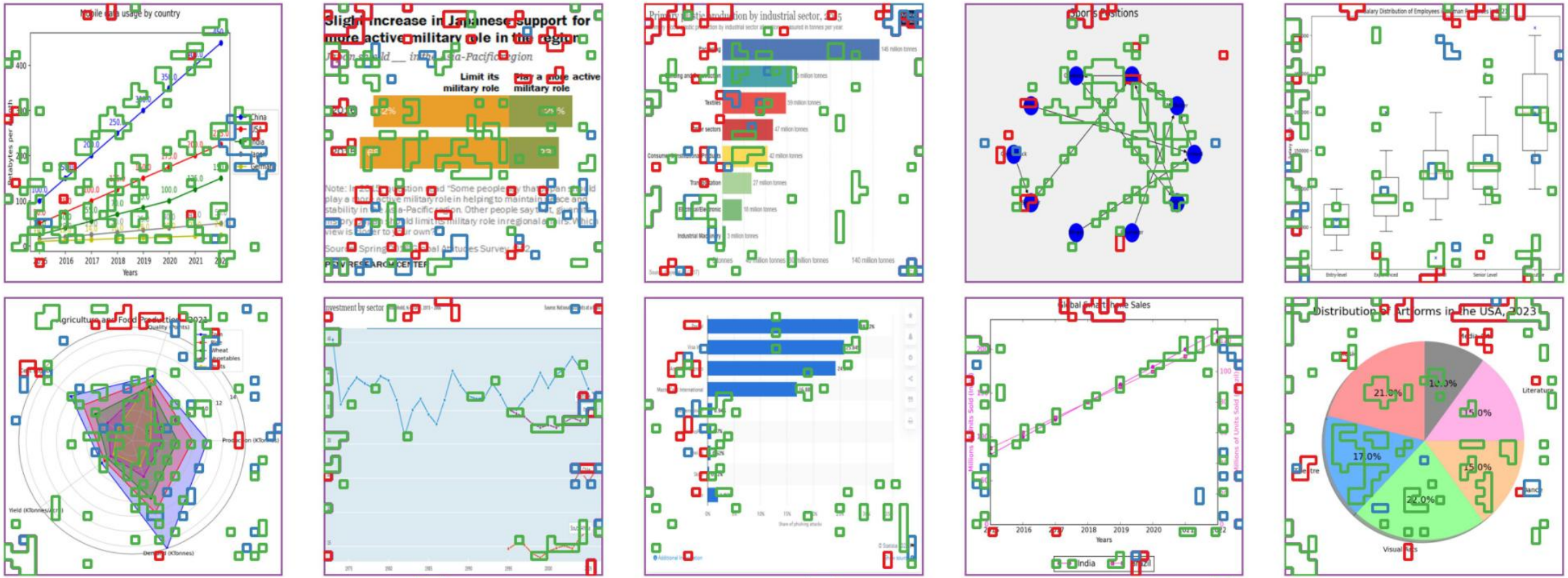
+**12.64**% on ChartQA@0.05



+**3.26**% on ChartBench_all

Qualitative Analysis

What exactly are the different experts **paying attention to**?

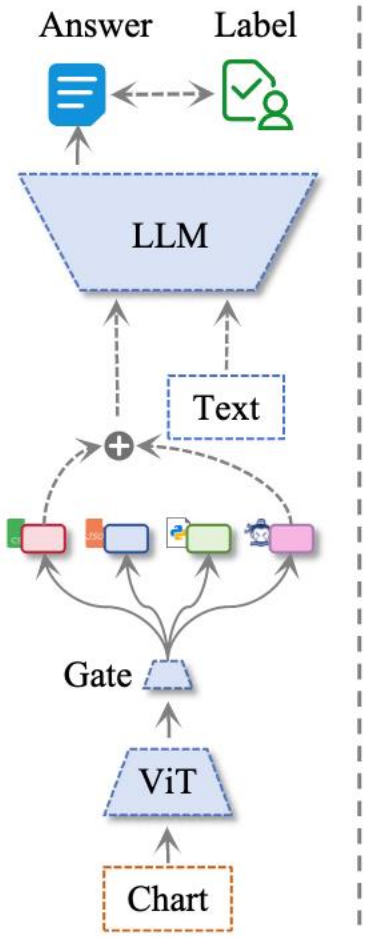


Expert initialized with InternlmXC-v2 Expert initialized with chart to table Expert initialized with chart to JSON Expert initialized with chart to code

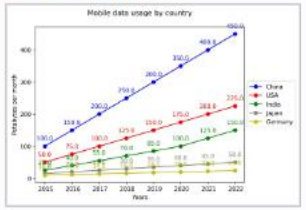
Figure 6: Visualizations of top-1 expert selection. Only the boundaries of the merged tokens are plotted.

Qualitative Analysis

What is ChartMoE **good at**?



Query Chart



Token-wise Top-1 Expert Selection in MoE Connector

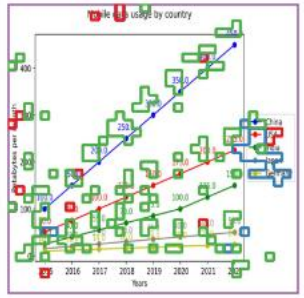


Chart QA

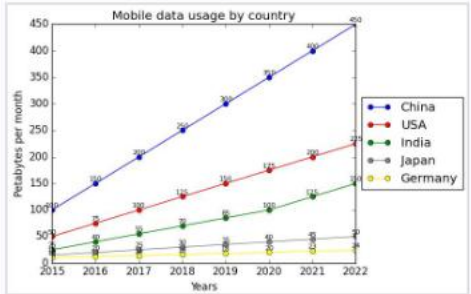
According to this chart, how much mobile data does China use in 2020?

It is arond 350 Petabytes per month.

Replot

Redraw the chart with python matplotlib code output. Pay attention to the line color!

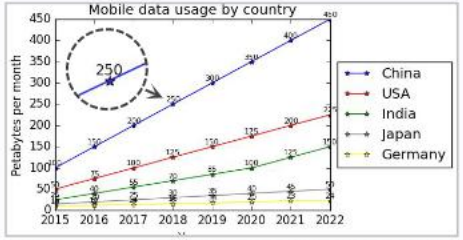
```
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use("classic")
data = {
'China': [100, 150, 200, 250, 300, 350, 400, 450],
'USA': [50, 75, 100, 125, 150, 175, 200, 225],
'India': [25, 40, 55, 70, 85, 100, 125, 150],
'Japan': [15, 20, 25, 30, 35, 40, 45, 50],
'Germany': [10, 12, 14, 16, 18, 20, 23, 24]
}
years = ['2015', '2016', '2017', '2018', '2019', '2020',
'2021', '2022']
df = pd.DataFrame(data, index=years)
plt.plot(df['China'], label='China', linewidth=1, marker='o',
color='#0000FF') # blue
plt.plot(df['USA'], label='USA', linewidth=1, marker='o',
color='#FF0000') # red
plt.plot(df['India'], label='India', linewidth=1, marker='o',
color='#008000') # green
plt.plot(df['Japan'], label='Japan', linewidth=1, marker='o',
color='#808080') # grey
plt.plot(df['Germany'], label='Germany', linewidth=1,
marker='o', color='#FFFF00')# yellow
for country in df.columns:
    for i, value in enumerate(df[country]):
        plt.text(i, value + 2, f'{value}', ha='center',
va='bottom', fontsize=8)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel('Years')
plt.ylabel('Petabytes per month')
plt.title('Mobile data usage by country')
plt.tight_layout(rect=[0, 0, 0.85, 1])
plt.show()
```



Editing

Redraw the chart with python matplotlib code output. 1) Pay attention to the line color! 2) Change the line marker to star!

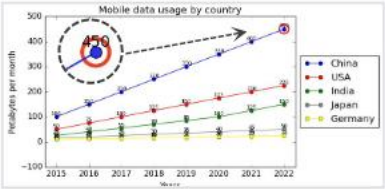
```
...
# Draw the chart, set the line width and mark points
plt.plot(df['China'], label='China', linewidth=1,
marker='*', color='#0000FF') # blue
plt.plot(df['USA'], label='USA', linewidth=1,
marker='*', color='#FF0000') # red
plt.plot(df['India'], label='India', linewidth=1,
marker='*', color='#008000') # green
plt.plot(df['Japan'], label='Japan', linewidth=1,
marker='*', color='#808080') # grey
plt.plot(df['Germany'], label='Germany', linewidth=1,
marker='*', color='#FFFF00')# yellow
...
```



Highlighting

Redraw the chart with python matplotlib code output. 1) Pay attention to the line color! 2) Circle the maximum value in red!

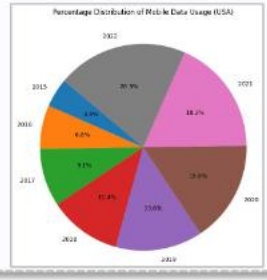
```
...
# get max value
max_value = df.max().max()
max_index = df.stack().idxmax()
country_name = max_index[1]
year_index = max_index[0]
# Circle max value in red
plt.scatter(year_index, max_value, s=200, edgecolor='red',
facecolor='none', linewidth=2)
...
```



Transformation

Plot the US data as a percentage pie chart in python.

```
import pandas as pd
import matplotlib.pyplot as plt
data = {
'China': [100, 150, 200, 250, 300, 350, 400, 450],
'USA': [50, 75, 100, 125, 150, 175, 200, 225],
'India': [25, 40, 55, 70, 85, 100, 125, 150],
'Japan': [15, 20, 25, 30, 35, 40, 45, 50],
'Germany': [10, 12, 14, 16, 18, 20, 23, 24]
}
years = ['2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022']
df = pd.DataFrame(data)
usa_data = df['USA']
plt.figure(figsize=(8, 8))
plt.pie(usa_data, labels=years, autopct='%1.1f%%', startangle=140)
plt.title('Percentage Distribution of Mobile Data Usage (USA)')
plt.show()
```



Thanks for listening!



We are on job market now.
Please **contact us**!