ChatGLM: An Alternative to ChatGPT

Jie Tang
KEG, Tsinghua University

Slides available at: http://keg.cs.tsinghua.edu.cn/jietang/ or Google Jie Tang
What is ChatGLM

- ChatGPT and GPT4 has gained enormous popularity
  - However, techniques behind GPT become a secret to all
- ChatGLM, an open-source ChatGPT alternative, toward unclosing the secret
  - GLM-130B: an open-source LLM base model
  - ChatGLM-6B: a lightweight open-source ChatGPT alternative
  - ChatGLM-130B: not open-sourced, but available through API

https://github.com/THUDM/GLM-130B
https://github.com/THUDM/ChatGLM3
ChatGLM-6B: An Open-Source Alternative

- ChatGLM-6B: 6.2B parameters, INT4 quantization (only need 6G memory)
- >50,000 stars on github
- >10,000,000 downloads on Huggingface
- No. 1 on Github Trending (2 week)
- No. 1 on Huggingface Trending (2 weeks)

https://github.com/THUDM/GLM-130B
https://github.com/THUDM/ChatGLM3

>600 Open-Sourced Apps developed based on ChatGLM
ChatGPT vs. ChatGLM

ChatGPT
DALL.E
Codex
WebGPT
GPT-4V

GPT
VS
GLM

ChatGLM
CogView
CodeGeeX
WebGLM
GLM-4V on the way (CogVLM, Agent...)
Who are you?

I am an AI assistant named ChatGLM, which is developed based on the language model jointly trained by Tsinghua University KEG Lab and Zhipu AI Company in 2023. My job is to provide appropriate answers and support to users' questions and requests.

Help me to write a blog outline which introduces ChatGLM

Sure, here's a blog outline that introduces ChatGLM:

I. Introduction
   A. Brief overview of ChatGLM
   B. Importance of AI assistants in today's world

II. How ChatGLM works
   A. Basics of AI language models

Welcome to try
Story generation

tell a 100 words story about athens in english, chinese, and greek

English: Athens, Greece's stunning capital, is a city of ancient ruins, modern architecture, and vibrant culture. Celebrated for its role in democracy, philosophy, and the Olympic Games, Athens offers visitors a unique blend of the past and present. Explore the Acropolis, Plaka, and Syntagma Square, or relax at one of the many beautiful parks and beaches. Enjoy mouthwatering Greek cuisine, from gyros to spanakopita, and immerse yourself in the city's lively arts and music scene. Athens is a destination where history comes alive and memories are made to last a lifetime.

中文：雅典，希腊美丽的首都，是一座融合了古老遗迹、现代建筑和生动文化的城市。雅典因其在民主、哲学和奥林匹克运动中的地位而闻名，游客可以在这里感受到独特的过去与现在的交融。游览Acropolis、Plaka和Syntagma广场，或者在众多美丽的公园和海滩放松身心，品尝令人垂涎的希腊美食，从烤肉串到菠菜派，让自己沉浸在城市繁忙的艺术和音乐氛围中。雅典是一个让历史焕发活力，留下终身难忘记忆的旅游胜地。

希腊语：Αθήνα, η καλύτερη και πόλη της Γαλλίας, είναι ένα σύγκρουσιμα της παλαιότερης και της σύγχρονης και της καλλιτέχνης μας. Στην Αθήνα θα βρείτε όλα τα σπουδάζοντα, τη σύγχρονη αρχιτεκτονική και την καλύτερη κοινωνία. Ξεκινήστε τη διαβόδεις την Ακρόπολη, τη Plaka και το Syntagma Square, ή καλύπτεστε στο ένα άλλο της πολλές όμορφες παρκυ και παραλίες. Γεύσατε την καλή Ποιοτικάδα κουζίνα, από τα γύρο στη στανακόπετα, και παραστήσετε σας ενώφεια μέσα στη σύγχρονη καλλιτέχνη και τη μουσική σκηνή της Αθήνας. Η Αθήνα είναι ένα σημείο όπου η ιστορία ζει στη ζωή και οι ευθυγράμματες της θεωρούνται η ζωή του κόσμου.
A company produces a toy for which the variable cost is 12.30 per unit and the fixed costs are 98,000. The toy sells for $17.98. Let \( X \) be the number of units produced and sold.

a) The total cost for a business is the sum of the variable cost and the fixed costs. Write the total cost \( C \) as a function of the number of units produced.

\[
C(X) = 12.30X + 98000
\]

b) The revenue \( R \) is the sales price minus the variable cost. Write the revenue \( R \) as a function of the number of units sold.

\[
R(X) = 17.98 - 12.30X
\]

c) The profit \( P \) is the revenue \( R \) minus the total cost \( C \). Write the profit \( P \) as a function of the number of units sold.

\[
P(X) = R(X) - C(X) = 17.98 - 12.30X - 98000
\]

Note: The negative sign indicates that the profit is a function of the number of units sold in the negative direction, meaning that as the number of units sold increases, the profit decreases.
Write a code for a bubble sort algorithm in Python, and give an array as an example.

Sure, here's a simple bubble sort algorithm implemented in Python:

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(n-i-1):
            if arr[j] > arr[j+1]:
                # Swap adjacent elements if they are in wrong order
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

To use this function, you'll need to define an array to sort, like so:

```python
arr = [64, 34, 25, 12, 22, 11, 90]
```

Once you have your array defined, you can call the `bubble_sort()` function on it, like so:

```python
sorted_arr = bubble_sort(arr)
```

Finally, you can print the sorted array like so:

```python
print(sorted_arr)
```

This should output something like:

```python
[11, 12, 22, 25, 34, 64, 90]
```

Let's take a look at an example array to better understand how the bubble sort algorithm works:

```python
arr = [5, 2, 9, 1, 5, 6, 10]
```

In this example, the first element (5) is sorted towards the bottom of the array, and the last element (10) is sorted towards the top. The middle elements (2, 5, and 9) are sorted in between their adjacent elements, as shown in the following sorted array:

```python
[1, 2, 5, 5, 6, 9, 10]
```

Overall, the bubble sort algorithm works by iterating through the array, comparing adjacent elements, and swapping them if they are in the wrong order. This process repeats until all elements in the array are sorted in increasing order.
In this picture, there are three football players. From left to right, they are Suárez, Neymar, and Messi.
“draw a dog with a hat”
大模型驱动的知识推理
1. 100B Base model
2. Supervised FT
3. RLHF
## General Language Model (GLM)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoregressive (GPT)</td>
<td>—</td>
<td>—</td>
<td>√</td>
</tr>
<tr>
<td>Autoencoding (BERT)</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Encoder-Decoder (T5)</td>
<td>—</td>
<td>√</td>
<td>—</td>
</tr>
<tr>
<td><strong>Autoregressive Blank-Infilling (GLM)</strong></td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

*Du and Qian et al. All NLP Tasks are Generation Tasks. ACL’22.*
General Language Model (GLM)

(a) Sample spans from the input text

Part A: $x_1 \ x_2 \ [M] \ x_4 \ [M]$

Part B: $x_5 \ x_6 \ x_3$

(b) Divide the input into Part A and Part B

(c) Generate the Part B spans autoregressively

(d) Self-attention mask

$$
\mathcal{L}_{GLM} = \mathbb{E}_{z \sim Z_m} \left[ \sum_{i=1}^{m} \sum_{j=1}^{l_i} - \log p \left( s_{z_i,j} | x_{corrupt}, s_{z<i}, s_{z<i, <j} \right) \right]
$$
General Language Model (GLM)
General Language Model (GLM)

Zeng, Liu, et al. GLM-130B: An Open Bilingual Pre-trained Model. ICLR’23
Results on Natural Language Understanding

• Better than BERT, T5, RoBERTa

Table 2. Results on the SuperGLUE dev set. Models with * are pre-trained for two times the number of steps of other methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>ReCoRD F1/Acc.</th>
<th>COPA Acc.</th>
<th>WSC Acc.</th>
<th>RTE Acc.</th>
<th>BoolQ Acc.</th>
<th>WiC Acc.</th>
<th>CB F1/Acc.</th>
<th>MultiRC F1a/EM</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT&lt;sub&gt;Base&lt;/sub&gt;</td>
<td>65.4/64.9</td>
<td>66.0</td>
<td>65.4</td>
<td>70.0</td>
<td>74.9</td>
<td>68.8</td>
<td>70.9/76.8</td>
<td>68.4/21.5</td>
<td>66.1</td>
</tr>
<tr>
<td>GLM&lt;sub&gt;Base&lt;/sub&gt;</td>
<td>73.5/72.8</td>
<td>71.0</td>
<td>72.1</td>
<td>71.2</td>
<td>77.0</td>
<td>64.7</td>
<td>89.5/85.7</td>
<td>72.1/26.1</td>
<td>70.7</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;Large&lt;/sub&gt;</td>
<td>76.3/75.6</td>
<td>69.0</td>
<td>64.4</td>
<td>73.6</td>
<td>80.1</td>
<td>71.0</td>
<td>94.8/92.9</td>
<td>71.9/24.1</td>
<td>72.0</td>
</tr>
<tr>
<td>UniLM&lt;sub&gt;Large&lt;/sub&gt;</td>
<td>80.0/79.1</td>
<td>72.0</td>
<td>65.4</td>
<td>76.5</td>
<td>80.5</td>
<td>69.7</td>
<td>91.0/91.1</td>
<td>77.2/38.2</td>
<td>74.1</td>
</tr>
<tr>
<td>GLM&lt;sub&gt;Large&lt;/sub&gt;</td>
<td>81.7/81.1</td>
<td>76.0</td>
<td>81.7</td>
<td>74.0</td>
<td>82.1</td>
<td>68.5</td>
<td>96.1/94.6</td>
<td>77.1/36.3</td>
<td>77.0</td>
</tr>
<tr>
<td>GLM&lt;sub&gt;Large (multi-task)&lt;/sub&gt;</td>
<td>80.2/79.6</td>
<td>77.0</td>
<td>78.8</td>
<td>76.2</td>
<td>79.8</td>
<td>63.6</td>
<td>97.3/96.4</td>
<td>74.6/32.1</td>
<td>75.7</td>
</tr>
<tr>
<td>GLM&lt;sub&gt;410M (multi-task)&lt;/sub&gt;</td>
<td>81.5/80.9</td>
<td>80.0</td>
<td>81.7</td>
<td>79.4</td>
<td>81.9</td>
<td>69.0</td>
<td>93.2/96.4</td>
<td>76.2/35.5</td>
<td>78.0</td>
</tr>
<tr>
<td>GLM&lt;sub&gt;515M (multi-task)&lt;/sub&gt;</td>
<td>82.3/81.7</td>
<td>85.0</td>
<td>81.7</td>
<td>79.1</td>
<td>81.3</td>
<td>69.4</td>
<td>95.0/96.4</td>
<td>77.2/35.0</td>
<td>78.8</td>
</tr>
<tr>
<td>T5&lt;sub&gt;Base&lt;/sub&gt;</td>
<td>76.2/75.4</td>
<td>73.0</td>
<td>79.8</td>
<td>78.3</td>
<td>80.8</td>
<td>67.9</td>
<td>94.8/92.9</td>
<td>76.4/40.0</td>
<td>76.0</td>
</tr>
<tr>
<td>T5&lt;sub&gt;Large&lt;/sub&gt;</td>
<td>85.7/85.0</td>
<td>78.0</td>
<td>84.6</td>
<td>84.8</td>
<td>84.3</td>
<td>71.6</td>
<td>96.4/98.2</td>
<td>80.9/46.6</td>
<td>81.2</td>
</tr>
<tr>
<td>BART&lt;sub&gt;Large *&lt;/sub&gt;</td>
<td>88.3/87.8</td>
<td>60.0</td>
<td>65.4</td>
<td>84.5</td>
<td>84.3</td>
<td>69.0</td>
<td>90.5/92.9</td>
<td>81.8/48.0</td>
<td>76.0</td>
</tr>
<tr>
<td>RoBERTa&lt;sub&gt;Large *&lt;/sub&gt;</td>
<td>89.0/88.4</td>
<td>90.0</td>
<td>63.5</td>
<td>87.0</td>
<td>86.1</td>
<td>72.6</td>
<td>96.1/94.6</td>
<td>84.4/52.9</td>
<td>81.5</td>
</tr>
<tr>
<td>GLM&lt;sub&gt;RoBERTa&lt;/sub&gt;</td>
<td>89.6/89.0</td>
<td>82.0</td>
<td>83.7</td>
<td>87.7</td>
<td>84.7</td>
<td>71.2</td>
<td>98.7/98.2</td>
<td>82.4/50.1</td>
<td>82.9</td>
</tr>
</tbody>
</table>
Results on Generation

- The most important thing is that **one model** can do all the things

<table>
<thead>
<tr>
<th>Model</th>
<th>RG-1</th>
<th>RG-2</th>
<th>RG-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASS</td>
<td>37.7</td>
<td>18.5</td>
<td>34.9</td>
</tr>
<tr>
<td>UniLM_{Large}</td>
<td>38.5</td>
<td>19.5</td>
<td>35.8</td>
</tr>
<tr>
<td>GLM_{Large}</td>
<td>38.6</td>
<td>19.7</td>
<td>36.0</td>
</tr>
<tr>
<td>GLM_{Large} (multi-task)</td>
<td>38.5</td>
<td>19.4</td>
<td>35.8</td>
</tr>
<tr>
<td>GLM_{410M} (multi-task)</td>
<td><strong>38.9</strong></td>
<td><strong>20.0</strong></td>
<td><strong>36.2</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Lambda (Accuracy)</th>
<th>BookWiki (Perplexity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLM_{Large} (uni)</td>
<td>0.0</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>GLM_{Large} (multi-task,uni)</td>
<td>47.4</td>
<td>15.1</td>
</tr>
<tr>
<td>- 2d positional encoding</td>
<td>45.8</td>
<td>15.1</td>
</tr>
<tr>
<td>GLM_{410M} (multi-task,uni)</td>
<td>49.5</td>
<td>14.5</td>
</tr>
<tr>
<td>GLM_{515M} (multi-task,uni)</td>
<td><strong>50.4</strong></td>
<td><strong>13.9</strong></td>
</tr>
<tr>
<td>GLM_{Large} (bi)</td>
<td>10.6</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>GLM_{Large} (multi-task,bi)</td>
<td>48.5</td>
<td>14.9</td>
</tr>
<tr>
<td>- 2d positional encoding</td>
<td>47.3</td>
<td>15.0</td>
</tr>
<tr>
<td>GLM_{410M} (multi-task,bi)</td>
<td><strong>53.5</strong></td>
<td><strong>14.3</strong></td>
</tr>
<tr>
<td>GLM_{515M} (multi-task,bi)</td>
<td><strong>54.9</strong></td>
<td><strong>13.7</strong></td>
</tr>
<tr>
<td>GPT_{Large} (uni)</td>
<td>50.1</td>
<td>14.4</td>
</tr>
</tbody>
</table>
Why 100B-scale model?

- What is 16 mod 12?
- 16 divided by 12 equals 1 remainder 4. So the answer is 4!

Why 100B-scale model?

Scaling Law

Scaling Law introduces complicated reasoning abilities

Model scale (# parameters in billions)
“Emergent abilities”

8 billion parameters
How to train a 100B–scale LLM?

- 8 months have witnessed numerous challenges

  - **Engineering:** How to train 100B-scale models from scratch?
    - Hygon DCU, NVIDIA A100, Ascend 910, Sunway
    - Frequent & random hardware failures, Megatron-DeepSpeed 3D pipeline, CUDA kernel efficiency, GPU memory overflow, 10K+ threads TCP init & comms…

  - **Algorithm:** How to stabilize the training of 100B-scale models?
    - The gradient norms of embeddings, Post-LN / Pre-LN stability, dataloader state seeds, computation precision in Softmax / Attention

Project
Conceived 2021.12

System Debug 2022.1

Data Large-Scale Tests 2022.2

Hygon, NVIDIA Ascend, Sunway 2022.3

Algo/Sys Tests 2022.4

Training Stability Issues 2022.5

Evaluations Quantization 2022.6

To be continued

http://keg.cs.tsinghua.edu.cn/glm-130b/
Training Stability of 100B-Scale Models

- Tradeoff: Stability (Slow) or Efficiency (Instable)

- Existing Solutions
  - OPT-175B: manually adjust LR & skip data when collapses (performance drop)
  - BLOOM 176B: embedding norm & BF16 (performance drop, few platform)
GLM-130B: Training Stability

- Attention score: Softmax in 32 to avoid overflow

\[
\text{softmax} \left( \frac{Q_i K_i^T}{\sqrt{d}} \right) = \text{softmax} \left( \left( \frac{Q_i K_i^T}{\alpha \sqrt{d}} - \max \left( \frac{Q_i K_i^T}{\alpha \sqrt{d}} \right) \right) \times \alpha \right) = \text{FP16} \left( \text{softmax} \left( \frac{Q_i K_i^T}{\alpha \sqrt{d}} \right) \times \alpha \right)
\]

Attention scores grow large --- exceeding FP16’s range

Zeng, Liu, et al. GLM-130B: An Open Bilingual Pre-trained Model. ICLR’23
Embedding Layer Gradient Shrink (EGS)

\[
\text{word\_embedding} = \text{word\_embedding} \times \alpha + \text{word\_embedding} \cdot \text{detach()} \times (1 - \alpha)
\]

Embedding Layer gradients can be magnitudes larger than others

(a) Gradient norm of embedding layer (left) and the first layer (right)  
(b) Training loss curves of GLM-40B with and without gradient shrink

Zeng, Liu, et al. GLM-130B: An Open Bilingual Pre-trained Model. ICLR’23
GLM-130B: Training Stability

- The final training run of GLM-130B

(c) GLM 130B’s experiments

(d) GLM 130B’s real training

Zeng, Liu, et al. GLM-130B: An Open Bilingual Pre-trained Model. ICLR’23
GLM-130B Training Lessons

2021.12
- The “千亿” (100B) project towards an open dense pre-trained GLM at 100B scale is conceived
- Survey pre-training strategies of existing models of similar scale, such as GPT-3, Gopher => Limited public info about how they were trained and issues they met
- Search for possible GPU clusters & sponsors

2022.1
- Test the performance of FP16/FP32 at 100B scale on one testing cluster
- Unexpected excessive memory usage in GLM => Torch is better with fixed length input sequences
- Inability to converge and try tricks from CogView and ViT => Use Sandwich-LN
- Frequent random hardware failures => Have to run HPCG test before each run

2022.2
- Very slow training speed than previously calculated => Optimize kernels and fuse operators => Find the input shape is critical to kernel performance
- Collect pre-training corpora and tokenize => Use icetk, the sentence piece is set to be the unigram mode
- Debug the 3D pipeline parallel in the newly-released Megatron and DeepSpeed

2022.3
- It can’t recover perfectly from optimizer states => Our customized dataloaders do not save its state seed properly in distributed training
- The memory per processor is too small => Require too many pipeline stages => Batch size is too large (up to 12,000) => Harm the model’s convergency
- It can’t learn more than 2,000 computing nodes => Overcome this and support 6,000-node training by tuning Linux kernel TCP parameters
- Collect data for multi-task instruction pre-training
- Receive opportunities to test trainings on several other clusters
- Very slow training speed than expected => The underlying element-wise operators don’t support fast computation on large-dimension vectors.

2022.4
- Optimize A100 kernel’s computing efficiency => A100 kernels prefer square-shaped inputs, and seq_len=2,048 is optimal for our hidden-state dimension (12,288)
- Inability to converge due to large gradient norm (170+) of input embeddings => Try embedding norm and gradient shrink, which turn out to be almost equivalent
- Naive post-LN or pre-LN diverges after several thousands of steps => Try Sandwich-LN with PB-Relax
- It still diverges after one week’s trial => The dataloader state seeds are not unified for different pipeline stages, resulting in a mismatch of input data and labels.
- Test two positional encodings: RoPE and Alibi => Alibi can be slower as it requires element-wise manipulation on attention matrices—changing num_heads *2,048 * 2,048 scalars per layer
- Test GeGLU and Gau => GAU converges faster with relatively poor performance on fine-tuned SuperGLUE
- Abnormal GPU memory usage of newly-added functions and classes => DeepSpeed hardcodes the function names for checkpoint activation
- Decode to train GLM with 130 billion parameters => allow inference on a DGX-A100 40G node

2022.5-6
- Implement a RoPE cuda operator in C++ => See unexpected precision errors and finally have it abandoned
- Sandwich-LN still diverges => 1) Reducing learning rate does not help; 2) Using Hinge cross-entropy becomes slower and harms performance; 3) Shifting to DeepNorm still diverges
- Use FP32 in softmax of attention => Success
- Find PB-Relax unnecessary for FP32 softmax => It also slows down training as it needs to manipulate the whole attention score matrices
- Experience few spikes in later training => 1) Reduce gradient shrink factor from 1 to 0.1: useful; 2) Reduce the learning rate: sometimes useful; 3) Jump the noisy data batches: sometimes useful
- Find a mistake in multi-task data after training for 20,000 steps => Use the correct data but it does not forget

2022.6-7
- Adapt the pipeline parallel checkpoints to ordinary parallel checkpoints for efficient inference on a single A100
- Work on evaluation scripts on datasets: MMLU, Big-bench, CLUE, SuperCLUE, etc.
- Implement: P-Tuning and P-Tuning v2 for parameter-efficient tuning on GLM-130B for tuning on SuperCLUE
- Work with BMinF on adapting GLM-130B to perform inference on a single V100 or 3090 => Use pipeline-style asynchronous swapping between main memory and GPU memory
- Try to fine-tune GLM-130B with fewer A100 nodes (i.e., 12-16 nodes) => Pipeline style fails due to too many pipeline stages => Find that data parallel can not be introduced for fine-tuning => Use 32-way model parallel for fine-tuning with reasonable performance

https://github.com/THUDM/GLM-130B
GLM-130B

English: better than GPT-3/OPT/PaLM on MMLU, LAMBADA, BIG-bench-lite

Aug., 2022-Mar. 2023, research use requests from ~1000 orgs in 70 countries

- Google
- Microsoft
- Facebook
- Stanford
- MIT
- UC Berkeley
- CMU
- Harvard
- Princeton
- Yale
- Cornell
- UIUC
- Cambridge
- Oxford
- Huawei
- Alibaba
- Tencent
- Baidu
- Meituan
- Bytedance
- Didi
- Xiaoice
- Xiaodu
- Xiaomi
- Xiaopeng
- Youdao
- Face++
- Ping An Cap
- Peking U.
- Zhejiang U.
- Shanghai JT U.
- Fudan U.
- USTC
- U of CAS
- Wuhan U.
- Naikai U.
- Hongkong U.
- CUHK
- HKUST
- BAAI
- Zhejiang Lab
- Shanghai AI Lab
**GLM-130B in HELM**

Stanford’s Holistic Evaluation of Language Models (HELM, Nov. 2022)

<table>
<thead>
<tr>
<th>Model</th>
<th>Model Creator</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1_Jumbo v1 (178B)</td>
<td>AI21 Labs</td>
</tr>
<tr>
<td>J1_Grande v1 (17B)</td>
<td>AI21 Labs</td>
</tr>
<tr>
<td>J1_Large v1 (7.5B)</td>
<td>AI21 Labs</td>
</tr>
<tr>
<td>Anthropic-LM v4-s3 (32B)</td>
<td>Anthropic</td>
</tr>
<tr>
<td>BLOOM (176B)</td>
<td>BigScience</td>
</tr>
<tr>
<td>T0++ (11B)</td>
<td>BigScience</td>
</tr>
<tr>
<td>Cohere large v20220609 (52.4B)</td>
<td>Cohere</td>
</tr>
<tr>
<td>Cohere large v20220720 (13.1B)</td>
<td>Cohere</td>
</tr>
<tr>
<td>Cohere medium v20220720 (6.1B)</td>
<td>Cohere</td>
</tr>
<tr>
<td>Cohere small v20220720 (410M)</td>
<td>Cohere</td>
</tr>
<tr>
<td>GPT-J (6B)</td>
<td>EleutherAI</td>
</tr>
<tr>
<td>GPT-NeoX (20B)</td>
<td>EleutherAI</td>
</tr>
<tr>
<td>T5 (11B)</td>
<td>Google</td>
</tr>
<tr>
<td>UL2 (20B)</td>
<td>Google</td>
</tr>
<tr>
<td>OPT (60B)</td>
<td>Meta</td>
</tr>
<tr>
<td>OPT (175B)</td>
<td>Meta</td>
</tr>
<tr>
<td>TNLG v2 (6.7B)</td>
<td>Microsoft/NVIDIA</td>
</tr>
<tr>
<td>TNLG v2 (530B)</td>
<td>Microsoft/NVIDIA</td>
</tr>
<tr>
<td>GPT-3 davinci v1 (175B)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>GPT-3 curie v1 (6.7B)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>GPT-3 babbage v1 (1.3B)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>GPT-3 ada v1 (350M)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>InstructionGPT davinci v2 (175B)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>InstructionGPT curie v1 (6.7B)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>InstructionGPT babbage v1 (1.3B)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>InstructionGPT ada v1 (350M)</td>
<td>OpenAI</td>
</tr>
<tr>
<td>Codex davinci v2</td>
<td>OpenAI</td>
</tr>
<tr>
<td>Codex curie v2</td>
<td>OpenAI</td>
</tr>
<tr>
<td>Codex babbage v1</td>
<td>OpenAI</td>
</tr>
<tr>
<td>Codex ada v1</td>
<td>OpenAI</td>
</tr>
<tr>
<td>YaLM (100B)</td>
<td>Yandex</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Model Creator</th>
</tr>
</thead>
</table>

GLM’s INT4 Weight Quantization Scaling Law
INT4 Quantization for RTX 3090s/2080s

- GLM-130B INT4 Quant. w/o perform. degradation

<table>
<thead>
<tr>
<th>Model Precision</th>
<th>GLM-130B</th>
<th></th>
<th>GPT-3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FP16</td>
<td>INT8</td>
<td>INT4</td>
<td>FP16</td>
</tr>
<tr>
<td>MMLU (acc, ↑)</td>
<td>44.75</td>
<td>44.71</td>
<td>44.80</td>
<td>43.9</td>
</tr>
<tr>
<td>LAMBADA (acc, ↑)</td>
<td>80.21</td>
<td>80.21</td>
<td>79.47</td>
<td>76.2</td>
</tr>
<tr>
<td>Pile (a part, BPB, ↓)</td>
<td>0.634</td>
<td>0.638</td>
<td>0.641</td>
<td>0.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GPU Type</th>
<th>128 Enc./Dec.</th>
<th>512 Enc./Dec,</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 × A100 (40G)</td>
<td>0.15s</td>
<td>0.18s</td>
</tr>
<tr>
<td>8 × V100 (32G)</td>
<td>0.31s</td>
<td>0.67s</td>
</tr>
<tr>
<td>4 × RTX 3090 (24G)</td>
<td>0.37s</td>
<td>1.30s</td>
</tr>
<tr>
<td>8 × RTX 2080 Ti (11G)</td>
<td>0.39s</td>
<td>1.04s</td>
</tr>
</tbody>
</table>
## GLM-130B

<table>
<thead>
<tr>
<th></th>
<th>Backbone</th>
<th>Training Objective</th>
<th>Quantization</th>
<th>Acceleration</th>
<th>Cross-Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPT3-175B</strong></td>
<td>GPT</td>
<td>SSL Only</td>
<td>—</td>
<td>—</td>
<td>NVIDIA</td>
</tr>
<tr>
<td><strong>OPT-175B</strong></td>
<td>GPT</td>
<td>SSL Only</td>
<td>INT8</td>
<td>Megatron</td>
<td>NVIDIA</td>
</tr>
<tr>
<td><strong>BLOOM-176B</strong></td>
<td>GPT</td>
<td>SSL Only</td>
<td>INT8</td>
<td>Megatron</td>
<td>NVIDIA</td>
</tr>
<tr>
<td><strong>GLM-130B</strong></td>
<td>GLM</td>
<td>SSL &amp; MIP</td>
<td>• INT8</td>
<td>• INT4</td>
<td>Faster Transformer</td>
</tr>
<tr>
<td><strong>Effects</strong></td>
<td>Performance Impro:</td>
<td></td>
<td></td>
<td></td>
<td>Diverse Supports:</td>
</tr>
<tr>
<td></td>
<td>• Big-bench-lite: +5.2%</td>
<td></td>
<td></td>
<td></td>
<td>It supports more diverse adoptions of LLMs</td>
</tr>
<tr>
<td></td>
<td>• LAMBADA: +2.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CLUE: +24.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• FewCLUE: +12.8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>**Affordable Serving:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>It saves 75% GPU mem in inference;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>It can infer on <strong>3090x4</strong> / <strong>2080x8</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Fast Inference:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7-8.4 faster than Pytorch;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5 faster than Megatron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Develop ChatGLM based on GLM-130B
Challenge 1: Simple NLP task vs. Complex task

- Simple NLP task => Complex task (e.g., logic reasoning)

**Math (GSM8k):**
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

**Last Letter Concatenation:**
Q: Take the last letters of the words in “Lady Gaga” and concatenate them.
Challenge 2: Static NLP vs. Dynamic knowledge

- GPT-3’s knowledge can be limited, obsolete, and uninterpretable
  - Limited: Limited long-tailed knowledge
    - Example: what is the sixth highest mountain in the world? *(Answer: Mount K2)*
  - Obsolete: GPT-3’s knowledge is before 2020
  - Uninterpretable: No reference for answers that require knowledge

**Case Study: Limited Knowledge**

What is the second highest mountain on earth?

The second highest mountain on earth is Cho Oyu which is in the Himalayas in Tibet, China and Nepal. Its height is 8,188 meters or 26,906 feet which is 28 feet taller than Mount Everest.

**Case Study: Obsolete Knowledge**

Who is the president of US in 2022?

Hillary Clinton
Challenge 3: Traditional NLP vs. Align with Human

- Case Study: Explain the moon landing to a 6 year old in a few sentences.
  - Without proper `prompt engineering`, GPT-3 and GLM-130B won’t return satisfying answers, either

<table>
<thead>
<tr>
<th>PROMPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explain the moon landing to a 6 year old in a few sentences.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMPLETION</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3</td>
</tr>
<tr>
<td>Explain the theory of gravity to a 6 year old.</td>
</tr>
<tr>
<td>Explain the theory of relativity to a 6 year old in a few sentences.</td>
</tr>
<tr>
<td>Explain the big bang theory to a 6 year old.</td>
</tr>
<tr>
<td>Explain evolution to a 6 year old.</td>
</tr>
</tbody>
</table>
Develop ChatGLM based on GLM-130B

Augmenting Code, Alignment, Web, Image understanding...

GLM-130B (ACL’22, ICLR’23)
Base model
~1TB, 4096

Text & Code

CodeGeeX (KDD’23)
Code → Reasoning
~0.4TB, 4096

Supervised Fine-tune

GLM-130B++ (GLM 3.5)
Instruction following

Web

WebGLM (KDD’23)
Combining Search

ChatGLM
Chat Product

Image understanding

Visual-ChatGLM (NeurIPS’21/22, ICLR’23)
Multi-modal

2021.12 2022.09 2022.12 2022.12 2023.05
CodeGeeX

- 6B/13B parameters, 100+ languages
- Support both Nvidia and 910A/B
- Free VSCode and JetBrains plugins

Generating over 10M lines codes

Optimization
- Operators (Layernorm/Gelu/BatchMatmul/Add)
- Auto search for optimizing matrix multiplication

Performance
- Improve 257% on Ascend 910A
- Trained with over 1,000 Ascend 910A
Relay Diffusion Model (RDM)

- RDM transfers a low-resolution image into an equivalent high-resolution one via blurring diffusion and block noise.
- RDM achieved state-of-the-art FID on CelebA-HQ and sFID ImageNet-256 (FID=1.87)!

https://github.com/THUDM/RelayDiffusion
“draw a dog with a hat”
CogVLM

- CogVLM connects pretrained language model and image encoder by a trainable visual expert model

(a) The input of visual language model
(b) The visual expert built on the language model
CogVLM

**Visual Math Problem Solving**

Q1: How to solve it?

CogVLM: The problem in the picture is a simple math problem, where a sliced green apple is equated to a question mark. Specifically, in the middle of the picture, there are three green apples placed side by side and three cardboard signs. The signs have black characters on them: a plus sign, an equal sign, and a question mark. The equation shown in the picture is: $2 + 1 = ?$

Q2: What's the final answer?

CogVLM: $2+1=3$

**Grounding Visual Question Answering**

Q1: What color are the clothes of the girl whose hands are holding flowers? Let's think step by step.

CogVLM: Find the flowers [[328,519,436,780]]. Check the girl holding it, got the girl [[116,609,450,927]]. Check the clothes of it, got the clothes [[134,309,419,852]]. The clothes [[134,309,419,852]] are blue. So the answer is blue.

Q2: What color are the jeans of the girl whose hands are not holding flowers? Let's think step by step.

CogVLM: Find the not(flowers) hands [[684,637,710,728]]. Check the girl with it, got the girl [[526,109,880,546]]. Check the jeans of it, got the jeans [[530,613,880,871]]. The jeans [[530,613,880,871]] are gray. So the answer is gray.

Achieved the best on 10+ benchmarks
who are in this picture

In this picture, there are three football players. From left to right, they are Suárez, Neymar, and Messi.
WebGLM = GLM + Search

Why do people try to maintain eye contact while communicating?

People try to maintain eye contact while communicating because it is a form of body language which can help improve communication and create a positive impression. Eye contact is a non-verbal form of communication which can help show that you are actively listening and paying attention\(^1\). It can also make others more likely to like and respect you, making it an important tool for building and maintaining relationships\(^2\). Furthermore, a lack of eye contact can negatively impact communication, making misunderstandings more likely to occur\(^3\).

[1] Why Eye Contact is Important in Communication

Eye contact is a type of body language that is extremely important during communication and conversation. Sometimes, our eyes and body language speak even more than words. Keeping eye contact with the person you are talking to shows that you are actively listening and paying attention.

[2] Why Eye Contact is Important during Conversation?
KG engine ready...

> get_relations(Hurricane Marie)
[meteorology.tropical_cyclone.category, meteorology.tropical_cyclone.tropical_cyclone_season]

> get_neighbors(Hurricane Marie, meteorology.tropical_cyclone.category)

variable #0

#### AGENT ####

Thought: I need to first find the relations of Hurricane Marie.

Action: get_relations(Hurricane Marie)

#### USER ####

Observation: [meteorology.tropical_cyclone.category, meteorology.tropical_cyclone.tropical_cyclone_season]

#### ROUND 2 ####

#### AGENT ####

Thought: I need to find the category of Hurricane Marie. To achieve this, I will use the weather.tropical_cyclone.category relation.

Action: get_neighbors(Hurricane Marie, meteorology.tropical_cyclone.category)

#### USER ####

Observation: variable #0, which are instances of meteorology.tropical_cyclone

#### ROUND 3 ####

#### AGENT ####

Thought: Now I need to find other...
AgentTuning: Enabling Generalized Agent Abilities For LLMs

Six agentInstruct trajectory datasets
- 1,866 high-quality CoTs

Agent Tuning Mix-training
- 20% AgentInstruct + 80% ShareGPT

codes & models: [http://github.com/THUDM/AgentTuning](http://github.com/THUDM/AgentTuning)
Main Results

In-domain dist
Significant improvement

Out-domain dist
Good generalization

Better generalization

+76%
+57%
+176%
ChatGLM-6B

- Download from Huggingface
  - git clone https://huggingface.co/THUDM/chatglm3
- Download demo
  - git clone https://github.com/THUDM/ChatGLM3
  - cd ChatGLM-6B
- Install demo
  - pip install gradio
  - python web_demo.py
- Run the demo
  - python cli_demo.py
- Install the api
  - pip install fastapi uvicorn
  - python api.py
- Run ChatGLM on your own MAC (w/ Apple Silicon)
  - model = AutoModel.from_pretrained("your local path", trust_remote_code=True).half().to('mps')
# Open LLM Research

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Stars</th>
<th>Updated</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChatGLM-6B</td>
<td>35,471</td>
<td>last week</td>
<td>An Open Bilingual Dialogue Language Model</td>
</tr>
<tr>
<td>ChatGLM2-6B</td>
<td>14,125</td>
<td>2 weeks ago</td>
<td>An Open Bilingual Chat LLM</td>
</tr>
<tr>
<td>GLM-130B</td>
<td>7,315</td>
<td>Jul 25</td>
<td>An Open Bilingual Pre-Trained Model (ICLR 2023)</td>
</tr>
<tr>
<td>CodeGeex</td>
<td>7,215</td>
<td>2 weeks ago</td>
<td>An Open Multilingual Code Generation Model (KDD 2023)</td>
</tr>
<tr>
<td>CodeGeex2</td>
<td>4,850</td>
<td>Aug 12</td>
<td>A More Powerful Multilingual Code Generation Model</td>
</tr>
<tr>
<td>ChatGLM3</td>
<td>4,635</td>
<td>5 hours ago</td>
<td>Series: Open Bilingual Chat LLMs</td>
</tr>
<tr>
<td>VisualGLM-6B</td>
<td>3,541</td>
<td></td>
<td>Chinese and English multimodal conversational language model</td>
</tr>
</tbody>
</table>

[GitHub Repository](https://github.com/THUDM)
Bigmodel.ai—API Platform

ChatGLM-Pro
- Powerful
- 0.002$ /1K Tokens
- High quality, Knowledge base, reasoning

ChatGLM
- Flexible
- 0.0002$ /1K Tokens
- Balanced effect and cost, news writing, abstract generation, vertical search

ChatGLM-Lite
- Fast
- 0.0001$ /1K Tokens
- High speed, lower cost, chatting, customer service, classification, extraction
What’s the next?
Abstraction and Reasoning

1. Francois Chollet. On the Measure of Intelligence. 2019
Abstraction and Reasoning
Abstraction and Reasoning
Generative Agent

- Generative agents: computational software agents that simulate believable human behavior
  - A “Westworld” with 25 agents; Auto-GPT; AgentGPT…

Introducing Superalignment

We need scientific and technical breakthroughs to steer and control AI systems much smarter than us. To solve this problem within four years, we're starting a new team, co-led by Ilya Sutskever and Jan Leike, and dedicating 20% of the compute we've secured to date to this effort. We're looking for excellent ML researchers and engineers to join us.
Summary

- GPT vs GLM
  - ChatGPT vs. ChatGLM
  - DALL.E vs. CogView
  - Codex vs. CodeGeeX
  - WebGPT vs. WebGLM
  - GPT-4V vs. GLM-4V (CogVLM, AgentTuning…)

- 2024-toward AGI
References

• Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. CodeGeeX: A Pre-Trained Model for Code Generation with Multilingual Benchmarking on HumanEval-X. KDD’23.


• Jing Zhang, Xiaokang Zhang, Daniel Zhang-Li, Jifan Yu, Zijun Yao, Zeyao Ma, Yiqi Xu, Haohua Wang, Xiaohan Zhang, Nianyi Lin, Sunrui Lu, Jie Tang, and Juanzi Li. GLM-Dialog: Noise-tolerant Pre-Training for Knowledge-grounded Dialogue Generation. KDD’23.

• Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130B: An Open Bilingual Pre-trained Model. ICLR’23.

• Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers. ICLR’23.


• Jifan Yu, Xiaohan Zhang, Yifan Xu, Xuanyu Lei, Xinyu Guan, Jing Zhang, Lei Hou, Juanzi Li, and Jie Tang. XDAI: A Tuning-free Framework for Exploiting Pre-trained Language Models in Knowledge Grounded Dialogue Generation. KDD’22.

• Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. ACL’21.


Thank you!

Many many collaborators from Tsinghua and Zhipu AI!

https://github.com/THUDM/