HiBD
High-Performance Big Data

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

HiDL
High-Performance Deep Learning

Can I Talk to my Supercomputer?
Conversational AI Interface for HPC Systems

Pouya Kousha, Hari Subramoni, DK Panda

The Network Based Computing Laboratory

The Ohio State University

http://mvapich.cse.ohio-state.edu/

Follow us on

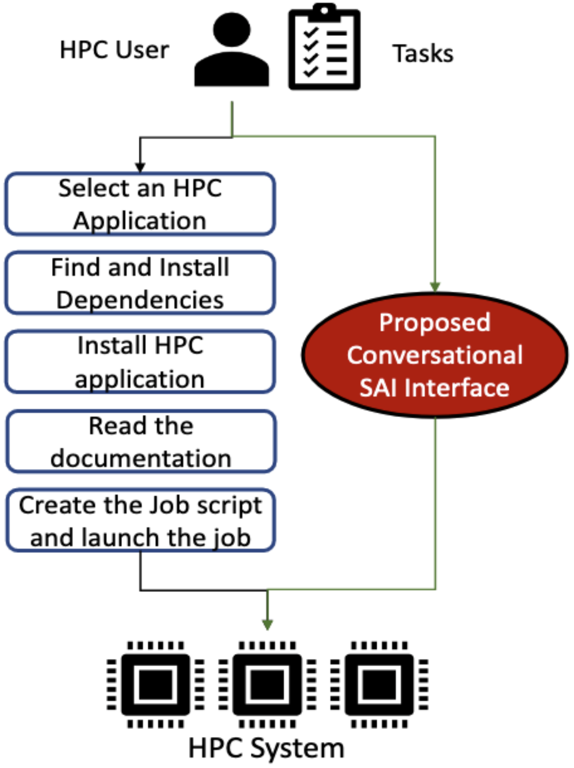
https://twitter.com/mvapich

Presentation Outline

- **Introduction and Motivation**
- Problem Statement
- Proposed Designs
- Performance Evaluation
- Demo
- Concluding Remarks

Introduction

- HPC usage is expanding: New users without HPC background
- Execution of parallel workloads is complex!
  - Learning new interfaces, features, and the terminologies
  - Complex dependencies, installation and execution
- Steep learning curves for executing tasks and utilizing HPC!
- Intuitive Expression: Users naturally convey needs through words & text
- Emergence of science gateways like Open OnDemand
  - GUI-based interface: more accessible and easier

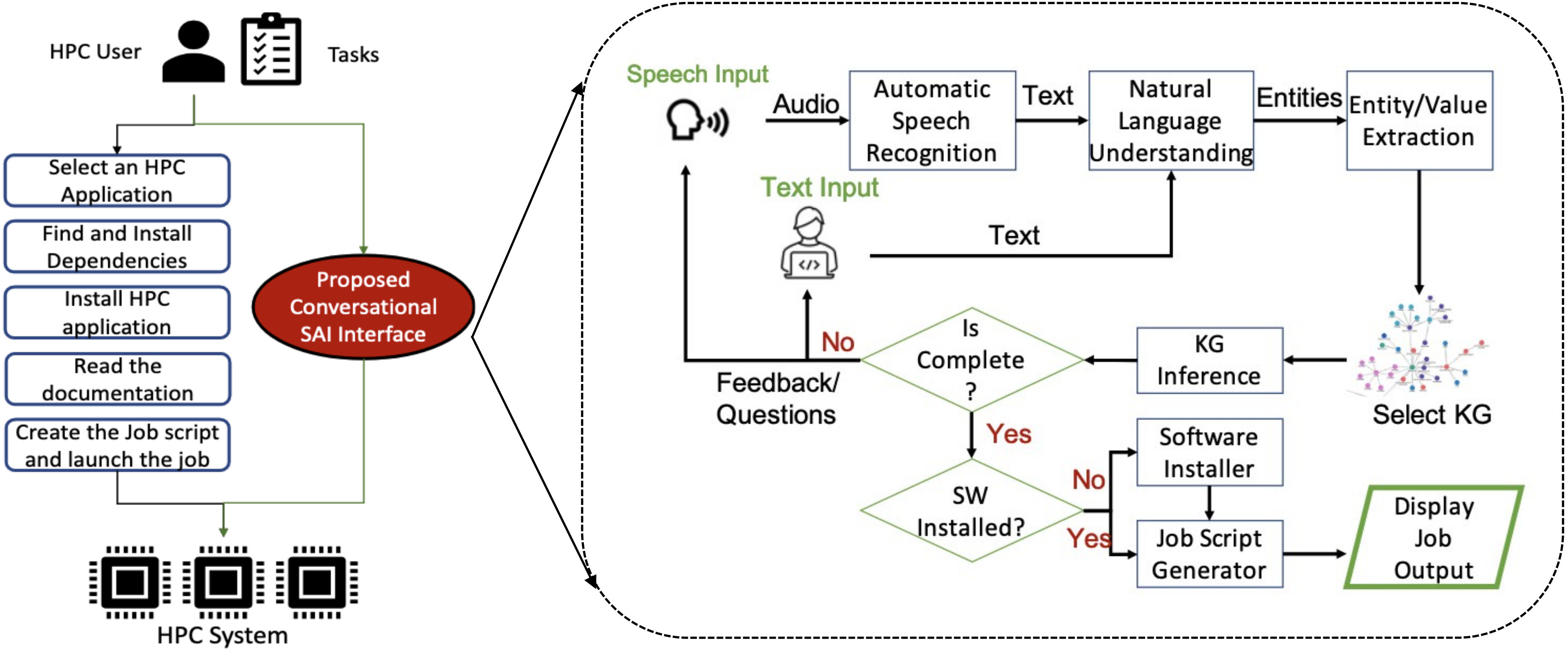


Research Challenges

Creating an AI-enabled conversational interface for HPC faces several challenges, including:

- Technical HPC terms: ASR and NLU models lack tailored HPC datasets, hindering accurate interpretation of technical terms and abbreviations.
- Complex relationships: Mapping complex HPC component relationships can be time-consuming and complex
- Software installation: HPC software package installation is a challenge, especially for novice users, even with package managers like Spack
- Conversational AI integration: Integrating conversational AI into science gateways requires a modular interface and determining the interface between conversational AI and science gateways

Proposed Framework for Conversational AI for HPC Tasks



Training Speech and Text Processing Models for HPC

Generating New HPC Dataset

Same as CAI but extra steps

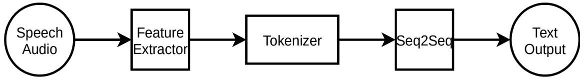
1. Basic queries and labels
2. Add combinations
3. Develop synonyms and mix
4. Include permutations

Five categories:

- System
- Software
- Model/Algorithm

Automatic Speech Recognition

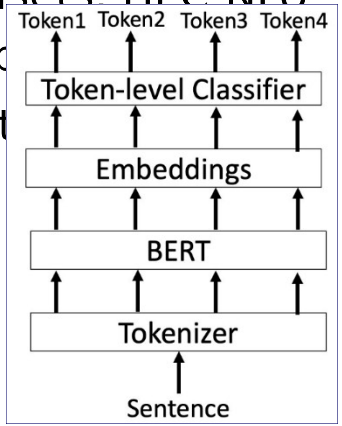
- Model: Speech2Text
- Dataset: TIMIT + HPC-ASR [proposed]
- Pretrained on LibriSpeech



Speech2Text architecture

Natural Language Understanding(NLU)

- Model: Bert entity recognition
- Datasets: HPC-NLU [prop
- Ident tasks



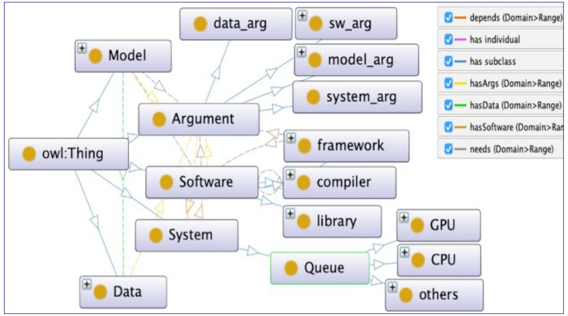
BERT architecture

Training Speech and Text Processing Models for HPC

New HPC Ontology

- Capture workload relations
- Create Knowledge Graph (KG) per application

Relation Property	Domain	Range	Description
canBe	any	any	Defines possible values (OR)
runs	any	Software or Model	Captures run capability
depends	Software	System	Captures software dependency
needs	any	any	Defines requirements (no default)
hasArgs	any	Argument	Defines optional values (defaults)
hasSoftware	any	Software	Captures software availability



Knowledge Graph Query

- Queries all KGs
- Max-hit KG selection
- Gathers needed/opt Args
- Check against user input
- Query user till completion

Software Installer:

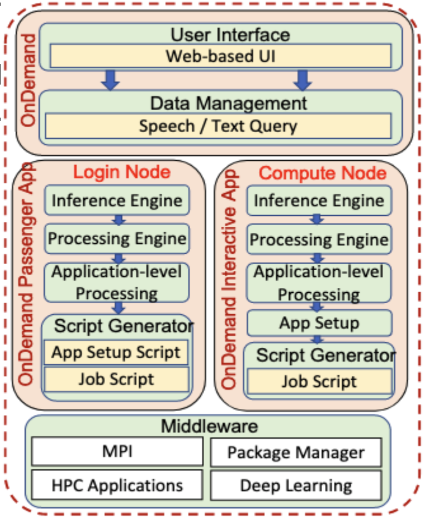
- Checks & installs dependencies via Spack
- Single Spack config and Env.
- Asynchronous installation

OnDemand Integration:

- HPC Integration and accessibility

Deployment modes:

- Passenger: shared r
- I r



DL Models Performance Evaluation

ASR Evaluation:

- Improved word error rate for Speech2Text model from 86.2% to 3.7%

NLU Evaluation:

- 60K training size, 5M test size
- Achieved 99% accuracy and precision

ASR + NLU Evaluation:

- Pipelines ASR+NLU for inference accuracy
- Testing 100 queries from 4 individuals
- **M1:** Adjust predicted sentence length to match original
- **M2:** Drop less important/incorrect words to match original content

Train Dataset	Test Dataset	WER
Base (LibriSpeech)	HPC-ASR	86.2
Base+TIMIT+HPC-ASR	HPC-ASR	**3.7**

Improved transition of speech to text

Test Dataset	F1-score	Precision	Recall
HPC-NLU (5M)	0.999	0.999	0.999

Predicting entities correctly!

Metric	User 1	User 2	User 3	User 4	**Average**
WER	10.3	8.6	8.3	4.9	**8.03**
Accuracy M2	0.97	0.90	0.80	0.95	**0.907**
Accuracy M1	0.84	0.81	0.83	0.92	**0.849**

User #1 and #4 are new to SAI and not used in training

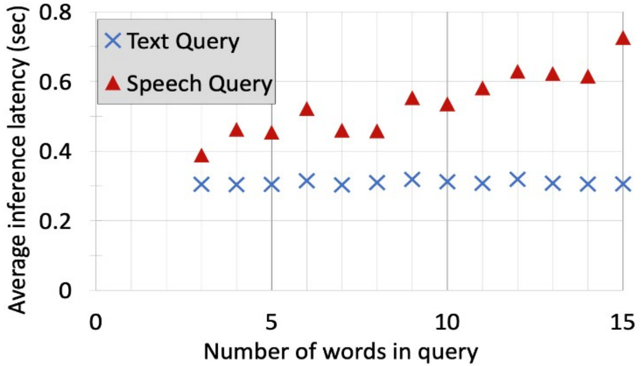
High success rate and flexibility recognizing new users' voices!

Source: Kousha et al, SAI: AI-Enabled Speech Assistant Interface for Science Gateways in HPC, ISC'23

Overhead of SAI Passenger App Pipeline for Different Queries

End-to-end Overhead: SAI Full Pipeline as passenger App

- Evaluate inference latency for various speech/text queries
- Exclude software installation and execution timing
- Speech latency increases with more query words
- Text latency remains constant



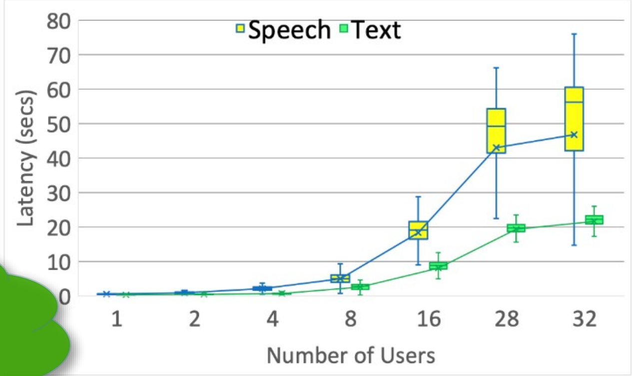
SAI passenger evaluation with different queries – avg 200 iterations

End-to-end Overhead of SAI Passenger App with Multi-Users

- Higher avg. latency for speech/text with more concurrent users
- Speech queries more affected than text queries

Login node performance degrades significantly with increased passenger mode usage

Deploy interactive app for smoother scaling



SAI passenger end-to-end latency across multiple users – 8 words text/speech and average of 200/100 iterations

Source: Kousha et al, SAI: AI-Enabled Speech Assistant Interface for Science Gateways in HPC , ISC'23

Interactive App deployment and Portability

Interactive App: Address performance degradation with scaling users

- Exclusive resources
- User-selected architecture

Observations:

- Lower latency on V100 GPU node
- Improved over passenger deployment

Architecture /Model	Deployment type	Total latency	ASR module	NLU module	KG module
BDW speech	Interactive	0.4919	0.23865	0.02275	0.22655
	Passenger	0.50245	0.2366	0.0217	0.2274
BDW text	Interactive	0.2665	N/A	0.0227	0.24335
	Passenger	0.27125	N/A	0.0218	0.24795
SKX speech	Interactive	0.44085	0.24105	0.0174	0.1754
SKX text		0.22095	N/A	0.0242	0.19585
V100 speech		0.40735	0.16585	0.0172	0.224
V100 text		0.2664	N/A	0.0225	0.2433
K80 text		0.2676	N/A	0.0225	0.2448

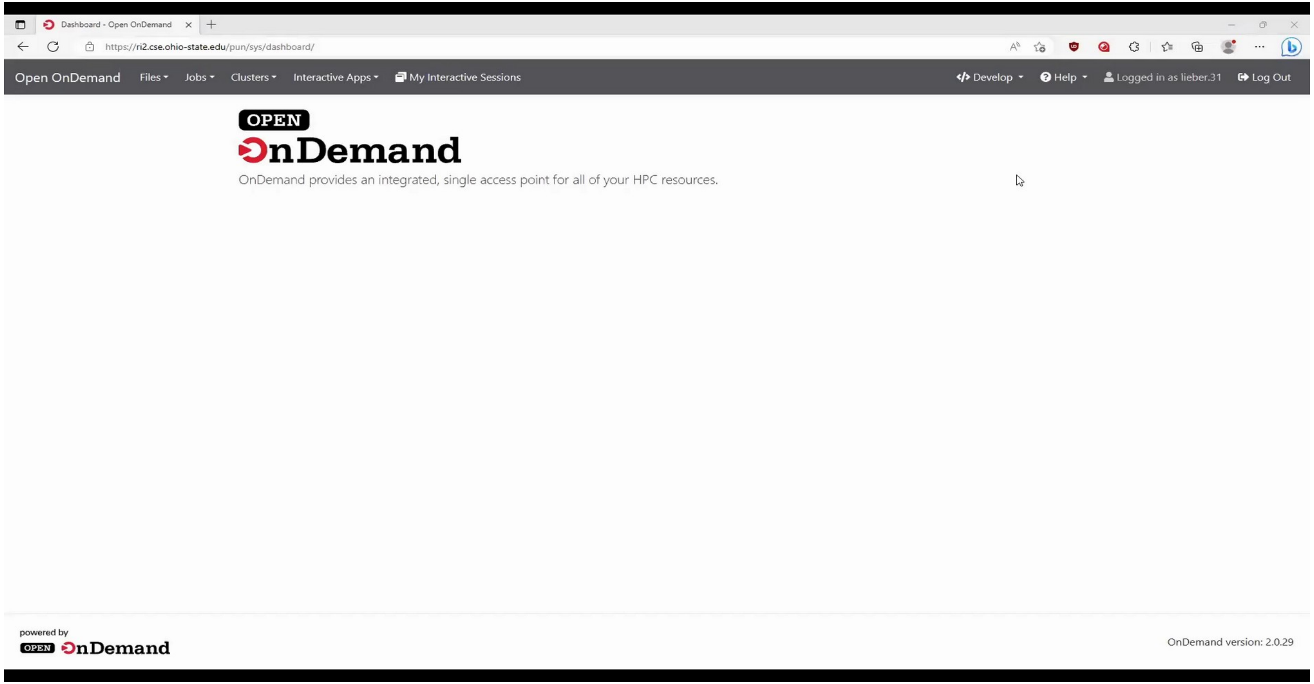
Compare breakdown & total latency on different architectures, 8-word text/speech query (100 speech/400 text iterations)

Extending SAI Support to New HPC Software

- Two-step process:
  - Create application KG using SAI-O ontology & supported relationships
  - Add application-specific terms to HPC-ASR and dataset
  - NLU is generic to detect new entities!
- SAI provides scripts for ASR model fine-tuning & NLU performance improvement
- Modular design: KG portability across systems, simplified deployment
- Integration with Open OnDemand for easy porting to new system architectures

Source: Kousha et al, SAI: AI-Enabled Speech Assistant Interface for Science Gateways in HPC , ISC'23

SAI Demo


Dashboard - Open OnDemand
https://ri2.cse.ohio-state.edu/pun/sys/dashboard/
Open OnDemand
Files
Jobs
Clusters
Interactive Apps
My Interactive Sessions
Develop
Help
Logged in as lieber.31
Log Out
OPEN OnDemand
OnDemand provides an integrated, single access point for all of your HPC resources.
powered by OPEN OnDemand
OnDemand version: 2.0.29

Concluding Remarks

- Proposed SAI, a conversational AI-enabled interface for science gateways in HPC, with Automatic Speech Recognition and Entity detection and classification model
  - Created an HPC speech and text dataset, defined a new ontology called SAI-O, and used knowledge graphs to check and validate user tasks, allowing for a general approach for any HPC application
  - Demonstrated capability by supporting three different HPC applications, and integrated SAI in Open OnDemand, deploying it on real HPC systems
  - Evaluated performance and functionality, with positive feedback from early users
- As future work we plan on releasing various components developed
  - HPC-ASR and HPC-NLU datasets
  - The retrained ASR and NLU models
  - Preform user survey

Follow us on

https://twitter.com/mvapich

Thank You!

Panda@cse.ohio-state.edu

Network Based Computing Laboratory

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

HiBD
High-Performance Big Data

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

HiDL
High-Performance Deep Learning

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/

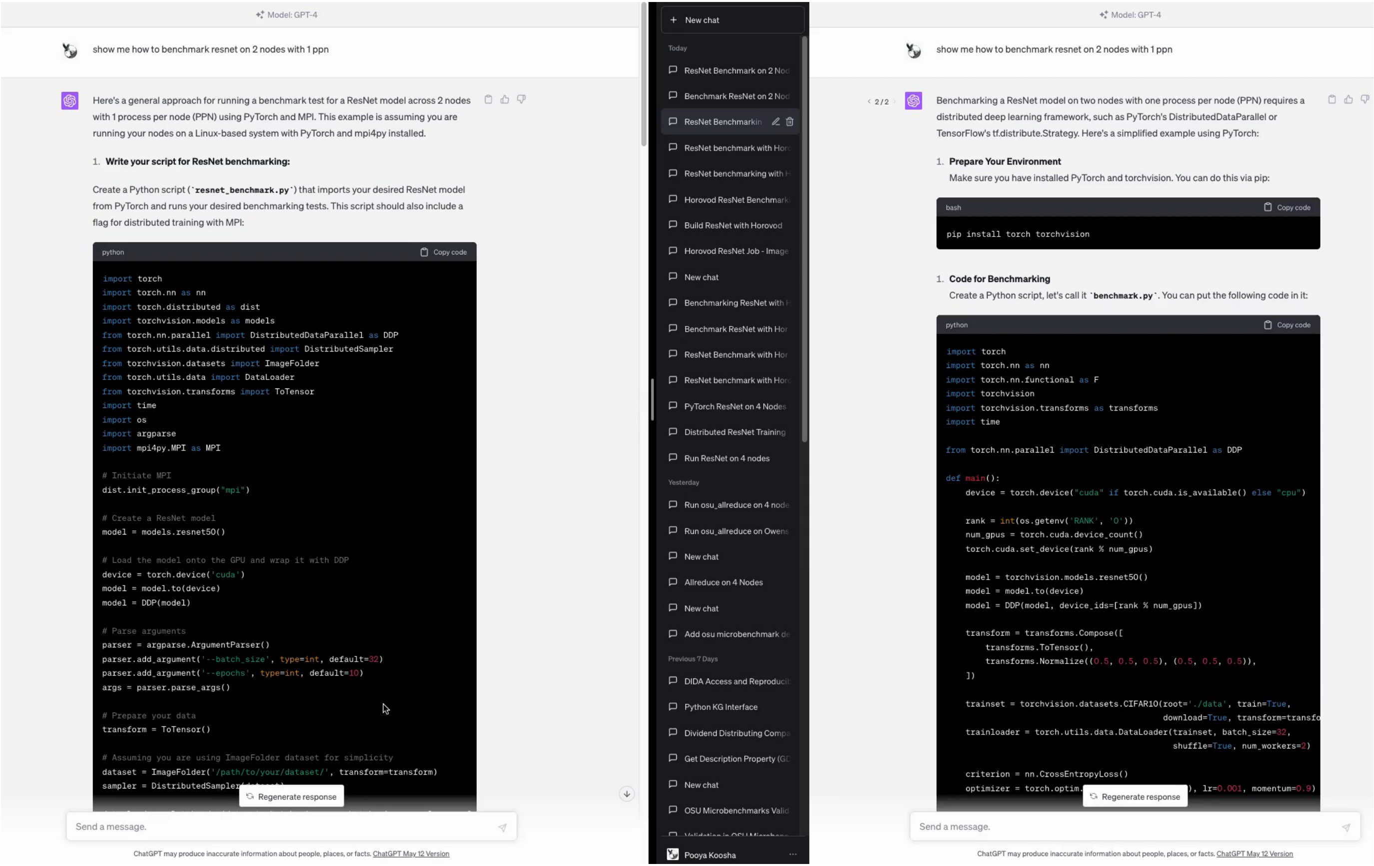
What about ChatGPT?

- ChatGPT is a general-purpose language model. It's versatile and it can handle a wide range of language tasks relatively well. However, it may not excel in any specialized task compared to a more dedicated tool
- ChatGPT can be very effective in communicating with humans, but it's far from the best when it comes to interacting with HPC systems
- Therefore, SAI is proposed and fine-tuned with the specific objective of effective and reliable interaction between both HPC users and HPC systems

VS

A Swiss army knife falls short if what we need is a multi-head screwdriver!

Comparison with State of the Art: ChatGPT



Comparison with State of the Art: ChatGPT (Cont.)

Qualitative Observations:

- Sensitivity to user inputs
- Lack of consistent answers
  - Same question had different answers
  - Can lead to reproducibility issues

Limitations

- ChatGPT sometimes writes plausible-sounding but incorrect or nonsensical answers. Fixing this issue is challenging, as: (1) during RL training, there's currently no source of truth; (2) training the model to be more cautious causes it to decline questions that it can answer correctly; and (3) supervised training misleads the model because the ideal answer depends on what the model knows, rather than what the human demonstrator knows.
- ChatGPT is sensitive to tweaks to the input phrasing or attempting the same prompt multiple times. For example, given one phrasing of a question, the model can claim to not know the answer, but given a slight rephrase, can answer correctly.

Courtesy: https://openai.com/blog/chatgpt

- Incorrect answers
  - Data is not distributed among GPUs!
  - Compilation issues

```
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=32,
                                          shuffle=True, num_workers=2)


criterion = nn.CrossEntropyLoss()
```

Screenshot of ChatGPT generated code – missing data distribution among processes

Comparison with State of the Art: ChatGPT (Cont.)

- ChatGPT produce irrelevant information
  - Provided code and extra parameters
  - Used CIFAR dataset without inquiring user

- Not best practice always!
  - Disabled InfiniBand for NCCL
  - No knowledge of best practices for performance

- Lack of specific HPC system configuration and knowledge

bash Copy code

```bash
#!/bin/bash

#SBATCH --job-name=pytorch_resnet     # Job name
#SBATCH --nodes=2                     # Number of nodes
#SBATCH --ntasks-per-node=2           # Tasks per node
#SBATCH --cpus-per-task=2             # CPUs per task, adjust as needed
#SBATCH --gres=gpu:1                  # Request for 2 GPUs per node
#SBATCH --mem=32G                     # Memory per node, adjust as needed
#SBATCH --time=10:00:00               # Time limit hrs:min:sec, adjust as ne
#SBATCH --output=/path/to/log_%j.log  # Log file with job number

# Initialize PyTorch distributed
source /path/to/your/conda/activate.sh
conda activate your_env

export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=^docker0,lo
export NCCL_IB_DISABLE=1  # Uncomment this if you are not using Infiniband

# PyTorch distributed
python -m torch.distributed.launch --nproc_per_node=2 --use_env your_pytorch
```

```bash
module load cuda/10.0
module load python/3.8.0
source ~/my_python_env/bin/activate

srun python -m torch.distributed.launch --nproc_per_node=1 --use_env ben
```

Screenshot of ChatGPT generated codes