

QCE23 Tutorial – 09/21/2023

QNLP on HPC+QC

In-Saeng Suh, Prasanna Date,
Antonio Perez, Kathleen Hamilton,
Mayanka Shekar, John Gounley and
Georgia (Gina) Tourassi

Oak Ridge National Laboratory

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

This research is supported by the Department of Energy Office of
Science Advanced Scientific Computing Research program office.



Happy Holidays!

0 2 ✓



Kothe, Douglas via CCSD via NCCS <nccs@email.ornl.gov>

To: ccsd@email.ornl.gov

Cc: Kothe, Douglas via CCSD <ccsd@email.ornl.gov>



Tue 12/27/2022 7:57 AM

I asked ChatGPT, the conversational AI language model making headlines, to “write a short Holiday poem for a staff of computer researchers.” Not bad in my opinion.

“‘Twas the night before Christmas and all through the lab,
Not a researcher was stirring, they were all on the job.
The computers were humming, the screens all aglow,
As the team worked on their projects, trying to make them go.

But as the night wore on, and the work had been done,
The team took a break and had some holiday fun.
They exchanged gifts and shared holiday cheer,
And took a moment to reflect on the past year.

So here's to the computer researchers, who work with such grace,
May your holiday season be filled with joy and good taste.
And from all of us, a heartfelt thank you,
For all that you do, and all that you'll do.”

Natural Language Processing (NLP)

- An area of study with elements from linguistics, computer science and artificial intelligence that focuses on the interaction between computers and human languages
 - The goal is to make computers capable of understanding text and spoken language in much the same way that human do
 - Machine translation, text summarization, chatbot creation, and spam detection, etc...
-
- The creation of quantum algorithm for NLP
- **Quantum Natural Language Processing (QNLP)**

MS6

Lambeq – a toolkit for QNLP

The screenshot shows the GitHub repository page for `lambeq`. The page includes the README.md file, a logo featuring a stylized letter 'A' inside a circle, and sections for build status, license, and contributors. The repository has 526 monthly downloads and is written in Python 99.9% of the time.

Build, lint and test package [passing](#) | license [Unable to select next Github token from pool](#) | pypi [v0.2.8](#) | downloads [526/month](#)

arXiv [2110.04236](#)

About

lambeq is a toolkit for quantum natural language processing (QNLP).

- Documentation: <https://cqcl.github.io/lambeq/>
- User support: lambeq-support@cambridgequantum.com
- Contributions: Please read [our guide](#).
- If you want to subscribe to lambeq's mailing list, let us know by sending an email to lambeq-support@cambridgequantum.com.

No packages published

Contributors 11

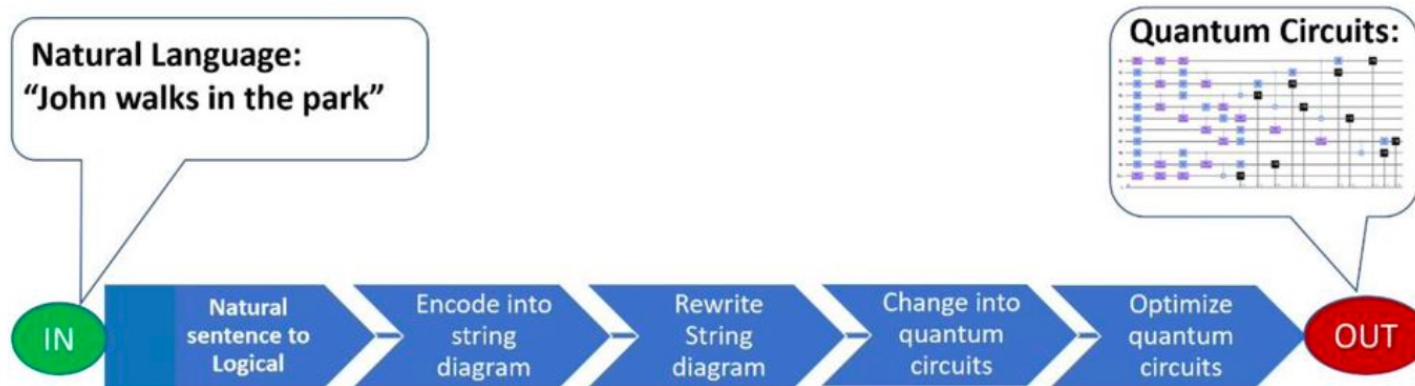
Languages

Python 99.9% | Shell 0.1%

MS3,6

Lambeq (after the late mathematician and linguist Joachim Lambek)

- The toolkit and Python library for QNLP
- It works with TKET quantum software development platform
- It automates tasks that are necessary for the large-scale implementation of QML pipelines designed in terms of compositional models of language



MS3,6

General Pipeline

Step 1: Syntax tree parse using Combinatory Categorial Grammar (CCG) parser

person prepares tasty dinner

$n \cdot (n^r \cdot s \cdot n^l) \cdot (n \cdot n^l) \cdot n \rightarrow (n \cdot n^r) \cdot$

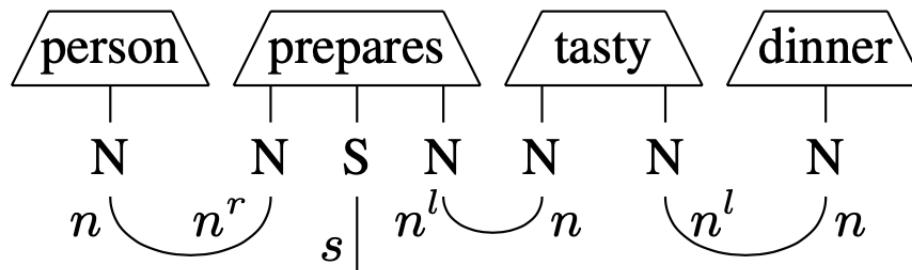
$s \cdot (n^l \cdot n) \cdot (n^l \cdot n) \rightarrow 1 \cdot s \cdot 1 \cdot 1 \rightarrow s$

n : noun

$n \cdot n^l$: adjectives

$n^r \cdot n \cdot n^l$: transitive verbs

Step 2: Construct the sentences' DisCoCat diagram



DisCoCat - Distributional Compositional Categorical

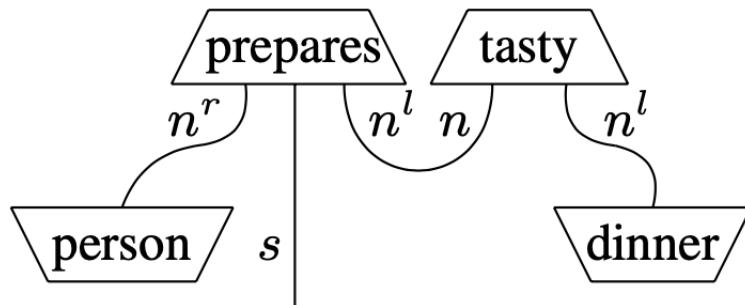
MS3,6

General Pipeline

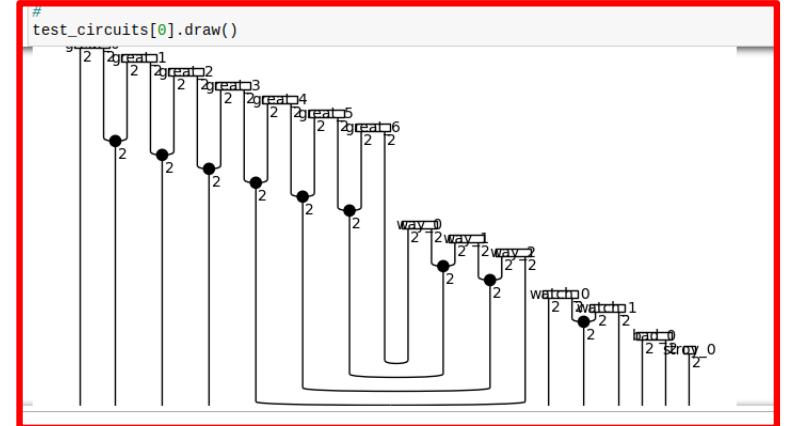
Step 3: Rewrite rules and remove cups (costly in quantum circuit)

→ Bending down all nouns

→ Reduced cups

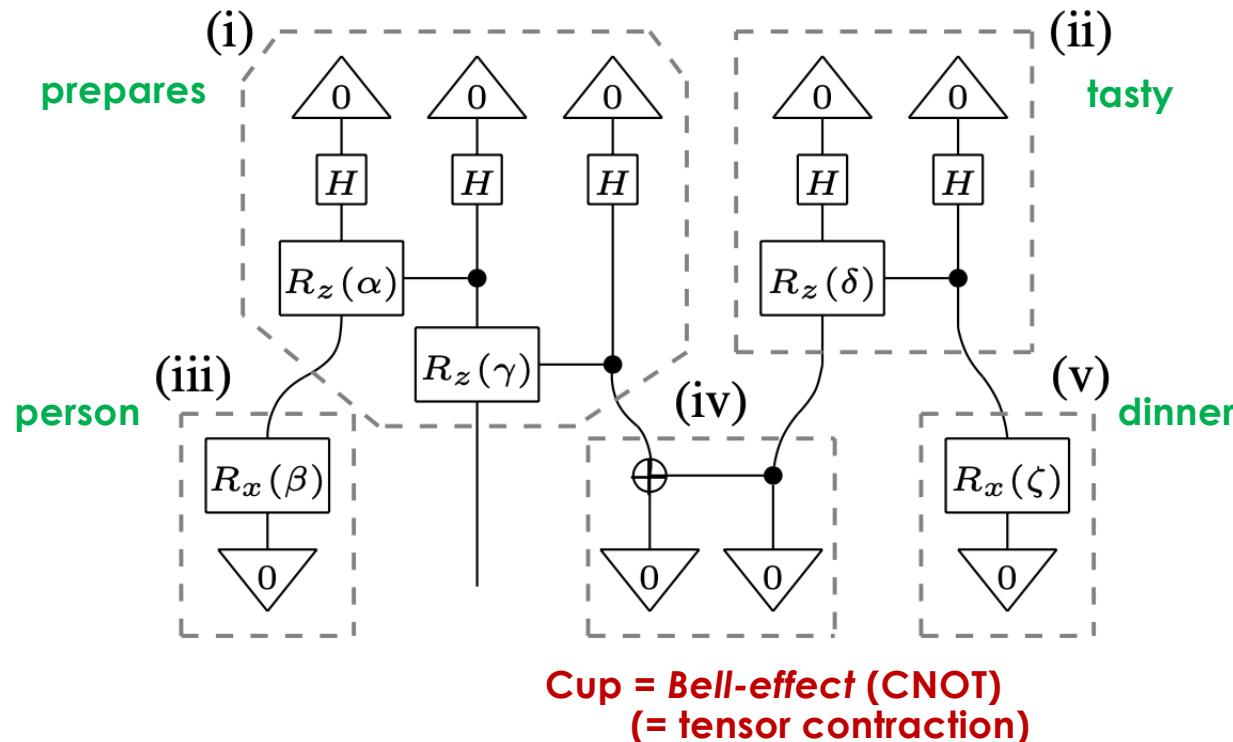


Example of 5 words sentence



General Pipeline

Step 4: DisCoCat diagram is mapped to a quantum circuit



MS3,6

General Pipeline

Step 5: A quantum compiler translates the quantum circuit into machine specific instructions

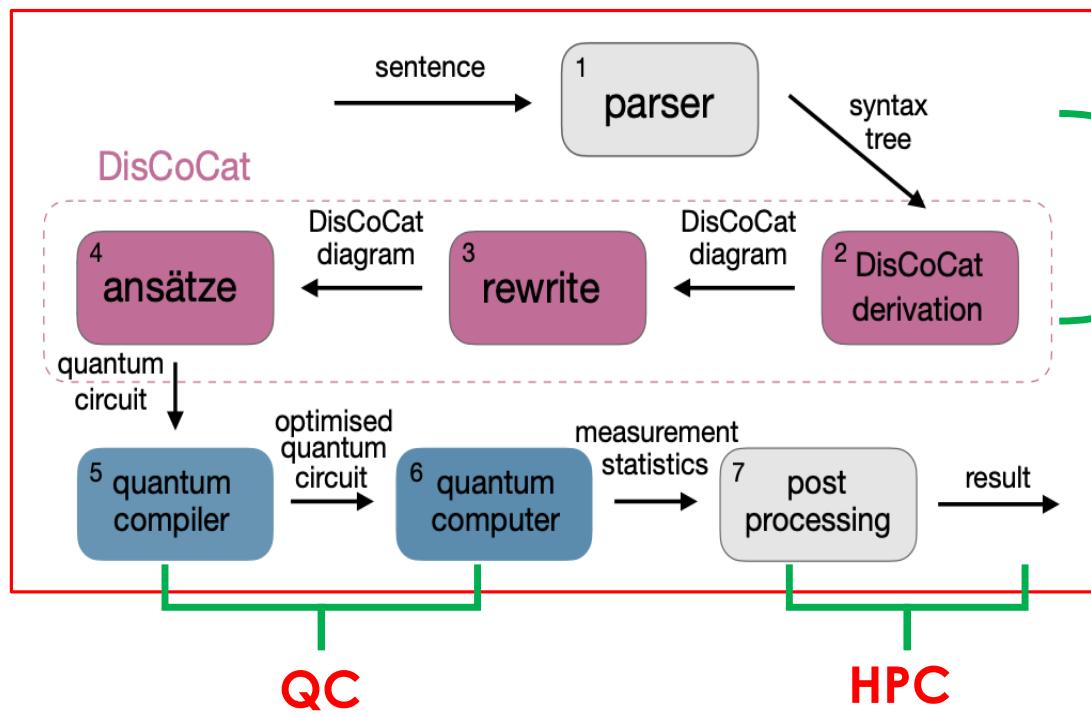
→ Circuit optimization

Step 6: Quantum computer runs the circuit n_{shots} time

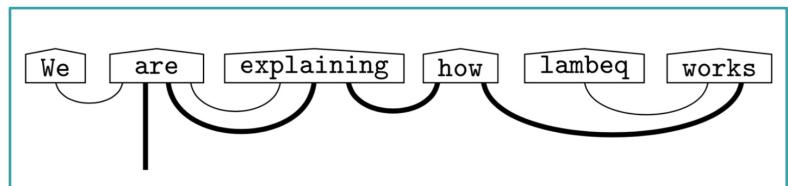
Step 7: Post-processing

MS3,6

General Pipeline diagram

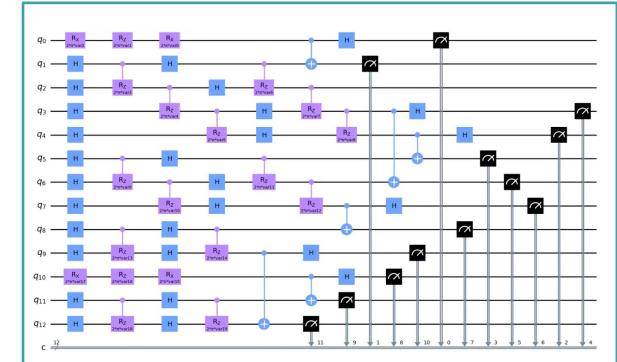


Example



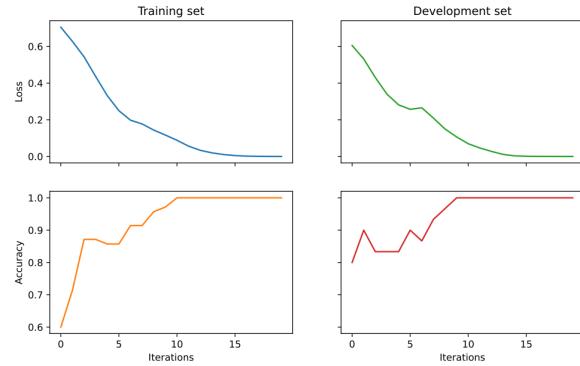
HPC

Lambeq

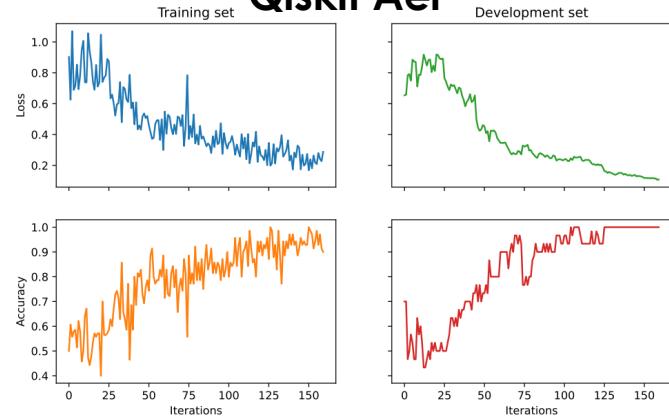


Results

Classical pipeline (PyTorch backend of TN)

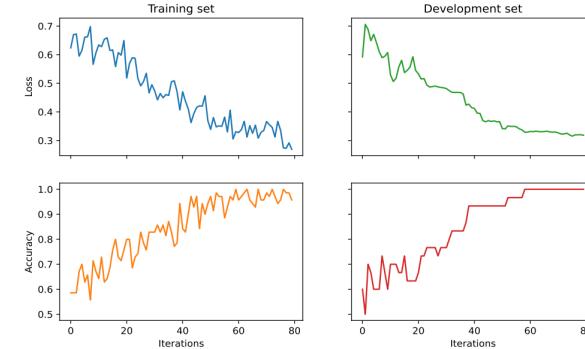


Noiseless simulation of quantum pipeline Qiskit Aer

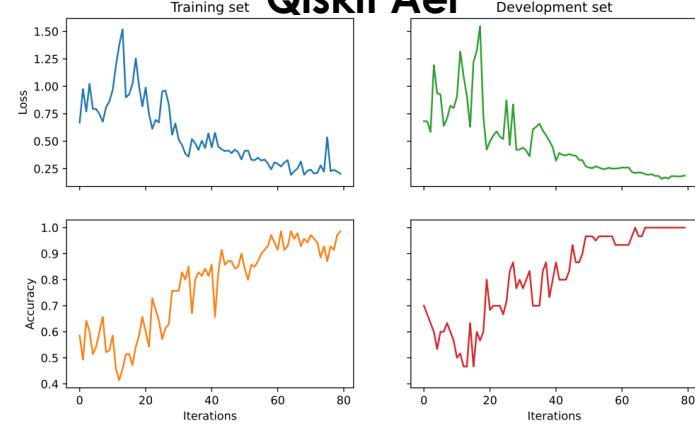


MS3,6

Classical simulation of quantum pipeline (with SPSA optimization)



Noisy simulation of quantum pipeline Qiskit Aer



Datasets

MS3,6

mc_dev_data.txt (30 lines)

```
0 skillful person prepares application .
1 man prepares tasty meal .
1 man prepares tasty sauce .
1 person bakes tasty meal .
0 man prepares useful program .
0 man runs program .
1 person prepares dinner .
1 man bakes sauce .
0 woman prepares software .
0 person prepares useful software .
0 skillful person debugs program .
0 person debugs application .
1 person cooks tasty dinner .
0 skillful person debugs software .
1 woman bakes tasty sauce .
0 skillful woman runs software .
1 skillful person prepares sauce .
0 person debugs useful program .
0 man runs application
```

MC (meaning classification) sentence data by CFG (Context-Free Grammar)

mc_train_data.txt (70 lines)

```
1 skillful man prepares sauce .
1 skillful man bakes dinner .
1 woman cooks tasty meal .
1 man prepares meal .
0 skillful woman debugs program .
1 woman prepares tasty meal .
0 person runs program .
0 person runs useful application .
1 woman prepares sauce .
1 woman prepares dinner .
1 skillful person prepares meal .
1 skillful person bakes dinner .
1 skillful woman bakes meal .
0 woman runs useful application .
1 man bakes tasty meal .
1 person prepares tasty meal .
0 woman runs application .
```

mc_test_data.txt (30 lines)

```
1 woman prepares tasty dinner .
1 woman cooks tasty sauce .
0 skillful woman prepares software .
1 skillful man prepares dinner .
1 skillful woman cooks sauce .
0 woman runs useful program .
0 skillful person runs software .
0 skillful person prepares program .
1 man prepares sauce .
1 person cooks tasty sauce .
1 man cooks sauce .
0 man prepares program .
0 skillful person prepares software .
1 woman bakes meal .
1 skillful man bakes sauce .
0 man prepares useful software .
0 woman debugs program .
0 skillful woman runs application .
0 man debugs software .
```

0 - IT
1 - Food

Running on Jupyter Notebook

Read in the data and create diagrams

```
In [9]: def read_data(filename):
    labels, sentences = [], []
    with open(filename) as f:
        for line in f:
            t = int(line[0])
            labels.append([t, 1-t])
            sentences.append(line[1:].strip())
    return labels, sentences

train_labels, train_data = read_data('datasets/mc_train_data.txt')
dev_labels, dev_data = read_data('datasets/mc_dev_data.txt')
test_labels, test_data = read_data('datasets/mc_test_data.txt')
```

```
In [10]: from lambeq import BobcatParser

parser = BobcatParser(verbose='text')

raw_train_diagrams = parser.sentences2diagrams(train_data)
raw_dev_diagrams = parser.sentences2diagrams(dev_data)
raw_test_diagrams = parser.sentences2diagrams(test_data)
```

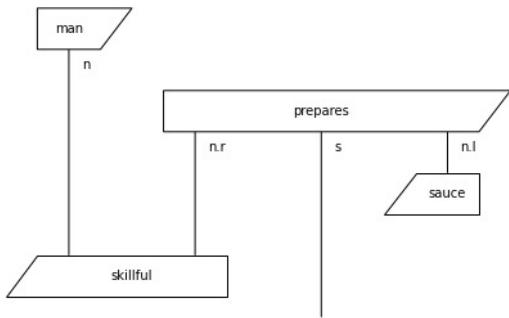
Tagging sentences.
Parsing tagged sentences.
Turning parse trees to diagrams.
Tagging sentences.
Parsing tagged sentences.
Turning parse trees to diagrams.
Tagging sentences.
Parsing tagged sentences.
Turning parse trees to diagrams.

Remove the cups

```
In [11]: from lambeq import remove_cups
```

```
train_diagrams = [remove_cups(diagram) for diagram in raw_train_diagrams]
dev_diagrams = [remove_cups(diagram) for diagram in raw_dev_diagrams]
test_diagrams = [remove_cups(diagram) for diagram in raw_test_diagrams]

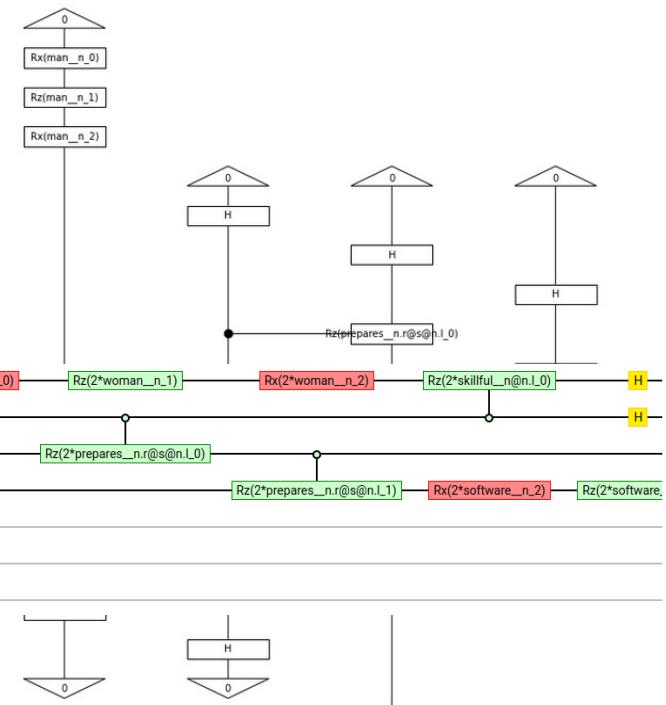
train_diagrams[0].draw()
```



Create circuits

MS3,6

```
In [12]: from lambeq import AtomicType, IQPAnsatz
ansatz = IQPAnsatz({AtomicType.NOUN: 1, AtomicType.SENTENCE: 1},
n_layers=1, n_single_qubit_params=3)
train_circuits = [ansatz(diagram) for diagram in train_diagrams]
dev_circuits = [ansatz(diagram) for diagram in dev_diagrams]
test_circuits = [ansatz(diagram) for diagram in test_diagrams]
train_circuits[0].draw(figsize=(9, 12))
```



MS3,6

Initialize trainer

```
In [16]: from lambeq import QuantumTrainer, SPSAOptimizer
        trainer = QuantumTrainer(
            model,
            loss_function=loss,
            epochs=EPOCHS,
            optimizer=SPSAOptimizer,
            optim_hyperparams={'a': 0.05, 'c': 0.06, 'A': 0.01*EPOCHS},
            evaluate_functions={'acc': acc},
            evaluate_on_train=True,
            verbose = 'text',
            seed=0
        )
```

```
In [17]: from lambeq import Dataset
        train_dataset = Dataset(
            train_circuits,
            train_labels,
            batch_size=BATCH_SIZE)
        val_dataset = Dataset(dev_circuits, dev_labels, shuffle=False)
```

Quantum simulator/device

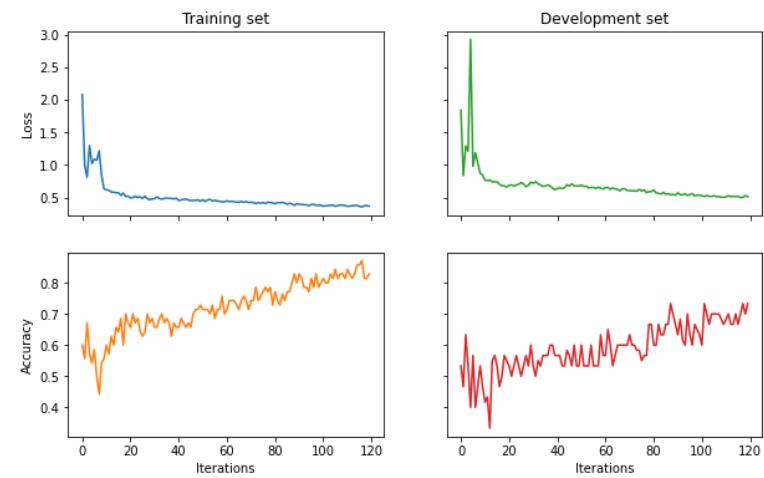
```
In [18]: #from pytket.extensions.qiskit import AerBackend
#        from pytket.extensions.qiskit import IBMQBackend
#
#        from lambeq import TketModel
#
#        all_circuits = train_circuits+dev_circuits+test_circuits
#
#        backend = AerBackend()
#        backend = IBMQBackend("ibmq_kolkata")
#        #backend = IBMQBackend("ibmq_lima")
#        #backend = IBMQBackend("ibmq_quito")
#        #backend = IBMQBackend("ibmq_manila")
#        #backend = IBMQBackend('ibm_washington')
#
#        #backend = IBMQBackend('ibmq_qasm_simulator')
#
#        backend_config = {
#            'backend': backend,
#            'compilation': backend.default_compilation_pass(2),
#            'shots': 8192
#        }
#        model = TketModel.from_diagrams(all_circuits, backend_config=backend_conf)
```

Train

```
In [18]: trainer.fit(train_dataset, val_dataset, logging_step=12)
        Job Status: job has successfully run
        Epoch 1:    train/loss: 0.9099    valid/loss: 0.9950    train/acc: 0.4286
        valid/acc: 0.3167
        Job Status: job has successfully run
        Job Status: job has successfully run
```

Post-processing

Test accuracy: 0.75



Running on Crusher and IBM Quantum

MS2

```
#backend = AerBackend()
backend = IBMQBackend('ibmq_manila')
backend_config = {
    'backend': backend,
    'compilation': backend.default_compilation_pass(2),
    'shots': 8192
}
model = TketModel.from_diagrams(all_circuits, backend_config=backend_config)
```

Quantum device

SLURM job script

```
#!/bin/bash
#SBATCH -A GEN008-qnlp
#SBATCH -J Crusher2IBM-01122023
#SBATCH -o %x-%j.out
#SBATCH -t 04:00:00
#SBATCH -p batch
#SBATCH -N 1
#
export PATH=/ccs/home/insaeng/anaconda3/x86_64/2022.10/bin:$PATH
#
export all_proxy=socks://proxy.ccs.ornl.gov:3128/
export ftp_proxy=ftp://proxy.ccs.ornl.gov:3128/
export http_proxy=http://proxy.ccs.ornl.gov:3128/
export https_proxy=https://proxy.ccs.ornl.gov:3128/
export no_proxy='localhost,127.0.0.0/8,*.ccs.ornl.gov'
#
python quantum_pipeline-sub-test-2-IBM.py
echo "~~~~~ test done ~~~~~"
```

Job submission to the Crusher compute node

MS2

```
(base) insaeng@login2:~/gpfs/crusher/QC/qnlp/lambeq-0.2.8/docs/examples$ sbatch quantum-pipeline-test-2-IBM.qsub
Submitted batch job 247140
(base) insaeng@login2:~/gpfs/crusher/QC/qnlp/lambeq-0.2.8/docs/examples$ more test_247140.out
```

Job monitoring on Crusher

```
(base) insaeng@login2:~/gpfs/crusher/QC/qnlp/lambeq-0.2.8/docs/examples$ squeue -u insaeng
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
      247140   batch  Crusher2  insaeng  R      10:43      1 crusher163
(base) insaeng@login2:~/gpfs/crusher/QC/qnlp/lambeq-0.2.8/docs/examples$ ls runs
Jan12_15-12-11_crusher163  Jan12_15-16-23_crusher021
```

Quantum pipeline Classical pipeline

Job monitoring on IBM Quantum

MS8,9

The screenshot shows the IBM Quantum Jobs interface. At the top, there's a search bar and a status filter set to 'All statuses'. Below the header is a table with columns: Job Id, Session Id, Status, Created, Run, Program, Compute resource, Provider, QR usage, and Tags. A red oval highlights the first row, which is pending. The other five rows are completed. A red box highlights the bottom section of the table, showing five completed jobs.

Job Id	Session Id	Status	Created	Run	Program	Compute resource	Provider	QR usage	Tags
63c06af9f7d85...		Pending ~ in about 5 hours till run	2 minutes ago		circuit-runner	ibmq_manila Queue position: 5	ibm-q/open/main Queue position: 5		⋮
63c06a95fc7ab...		Completed	3 minutes ago	2 minutes ago	circuit-runner	ibmq_manila	ibm-q/open/main		⋮
63c0699efc7ab...		Completed	7 minutes ago	4 minutes ago	circuit-runner	ibmq_manila	ibm-q/open/main		⋮
638fb19f94981...		Completed	about 1 month ago	about 1 month ago		simulator_statevector	ibm-q-ornl/ornl/csc500		⋮
638fa25a92fd...		Completed	about 1 month ago	about 1 month ago		ibmq_qasm_simulator	ibm-q-ornl/ornl/csc500		⋮
6384d81652d5...		Completed	about 2 months ago	about 2 months ago		ibmq_qasm_simulator	ibm-q-ornl/ornl/csc500		⋮

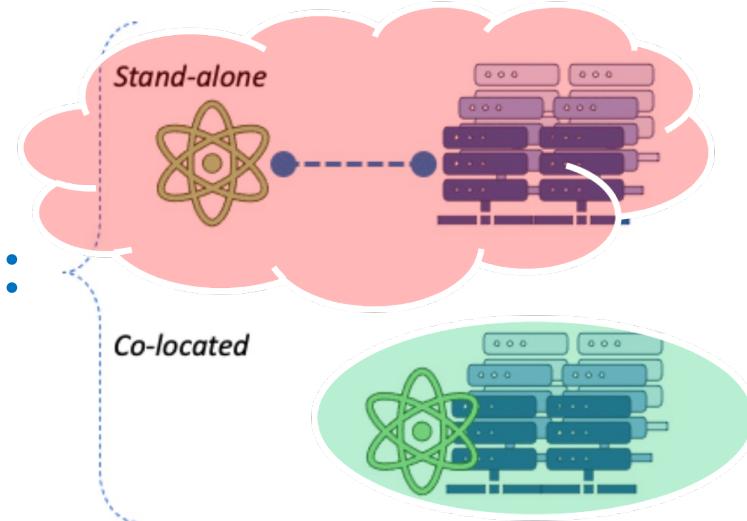
Job Id	Session Id	Status	Created	Run	Program	Compute resource	Prov
63c0a12808e6...		Completed	about 2 hours ago	about 2 hours ago	circuit-runner	ibmq_manila	ibm-q/op
63c0a0d508e6...		Completed	about 2 hours ago	about 2 hours ago	circuit-runner	ibmq_manila	ibm-q/op
63c0a08b1002...		Completed	about 2 hours ago	about 2 hours ago	circuit-runner	ibmq_manila	ibm-q/op
63c0a02dfc7ab...		Completed	about 2 hours ago	about 2 hours ago	circuit-runner	ibmq_manila	ibm-q/op
63c09fd4aabed...		Completed	about 2 hours	about 2 hours ago	circuit-runner	ibmq_manila	ibm-q/op

MS8,9

System Integration with Quantum

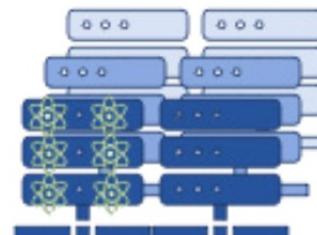
T. Humble and K. Britt 2016, Software systems for High-performance Quantum Computing
K. Britt and T. Humble 2017, High-Performance Computing with Quantum Processing Units
K. Britt, F. Mohiyaddin, and T. Humble 2017, Quantum Accelerators for HPC Systems

- **Loose integration model:**



- **Tight integration model:**

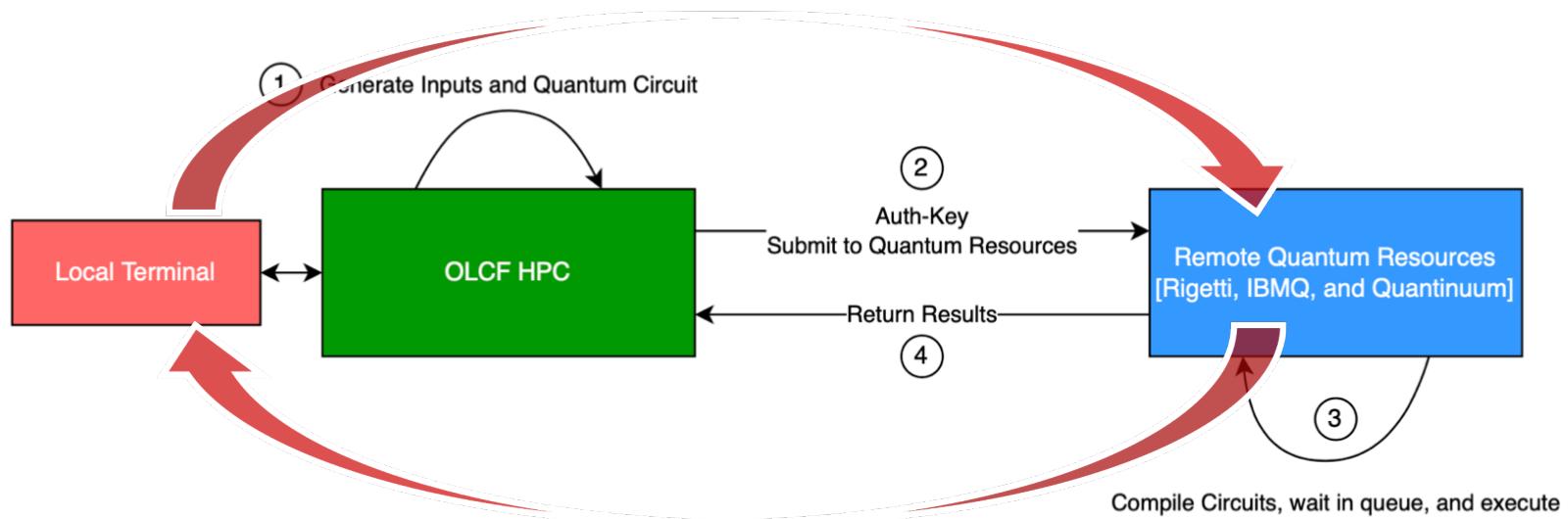
On-chip QPU



(Quantum for HPC, Bartsch, et al. 2021)

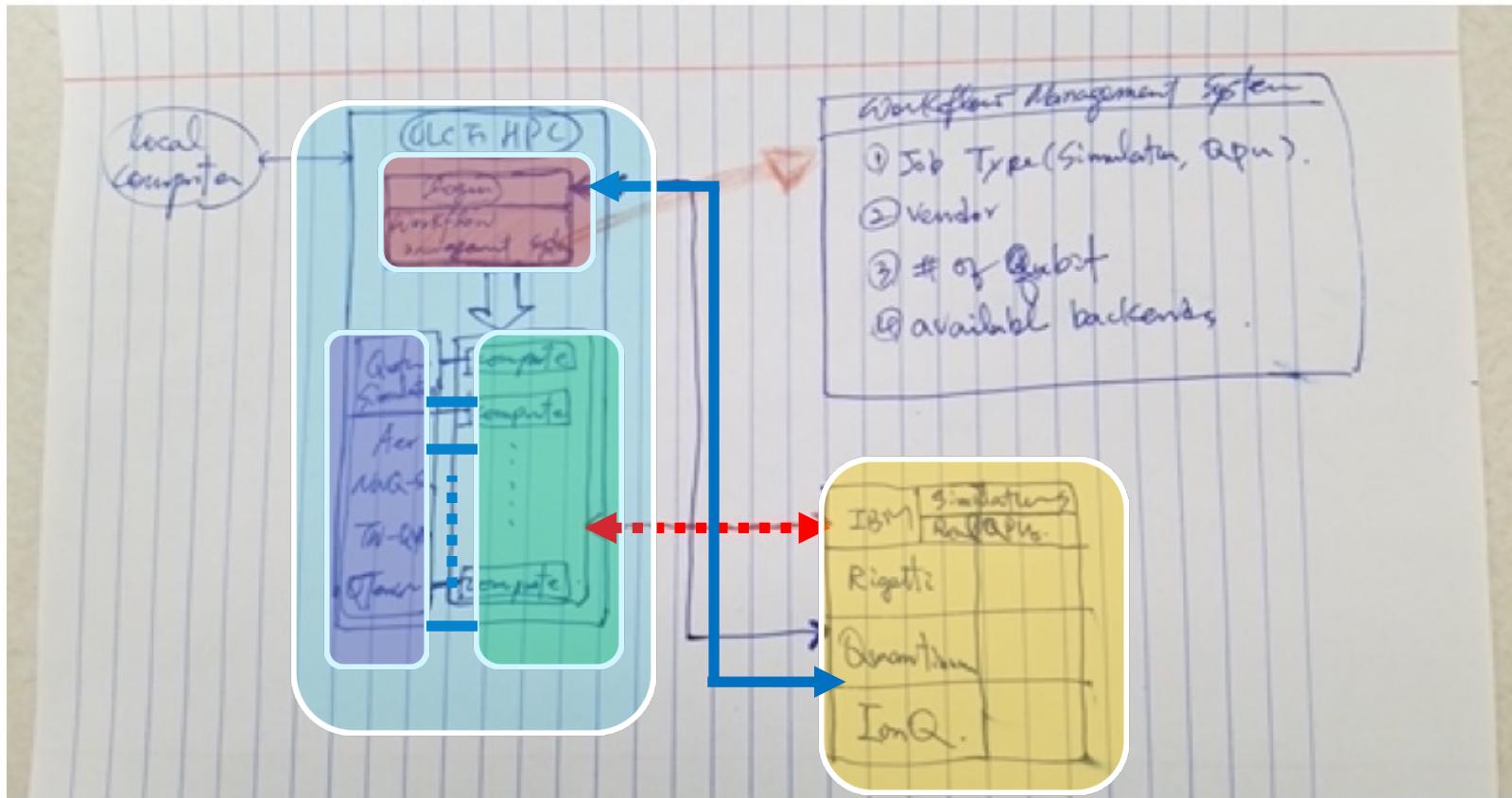
MS8,9

Workflows between OLCF HPC and Quantum Devices



Workflows between HPC and Quantum Computers, Bieberich, et al. ([ORNL](#)), 2022

Updated OLCF HPC - Quantum Workflow Schema^{MS9}



Thank you

Questions?