Quantum NLP with lambeq

Dimitri Kartsaklis Ian Fan Richie Yeung Thomas Hoffmann Vid Kocijan
Charles London Anna Pearson Robin Lorenz Alexis Toumi
Giovanni de Felice Konstantinos Meichanetzidis Stephen Clark
Bob Coecke

Cambridge Quantum / Quantinuum
17 Beaumont Street, Oxford, OX1 2NA, UK
`{firstname.lastname}@cambridgequantum.com`

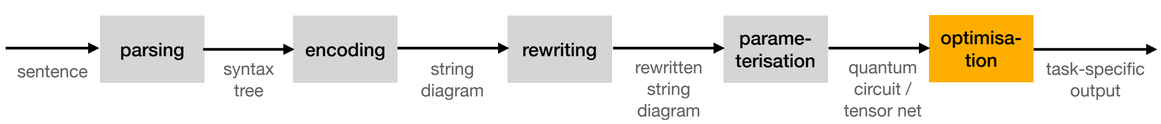
We present `lambeq`, the first high-level Python library for Quantum Natural Language Processing (QNLP). The open-source toolkit offers classes to implement all stages of an experimental QNLP pipeline, from text to string diagrams, tensor networks, and quantum circuits ready to be used on quantum hardware. `lambeq` supports syntactic parsing, rewriting of string diagrams, ansatz manipulation, and compositional models for preparing quantum-friendly representations of sentences. We present the architecture and modules in detail, demonstrating usage with illustrative examples, as well as testing the toolkit in practice with a simple NLP experiment to show that it is ready for use in real-world quantum settings.

1 Introduction

Quantum Natural Language Processing (QNLP) is a rapidly-growing area of research that explores how approaches to natural language tasks can utilise quantum phenomena, such as superposition, entanglement and interference. Current NISQ[1] computers allow researchers to run simple experiments on quantum hardware [15, 12, 14], but it requires a lot of low-level work to prepare even simple QNLP models.

To facilitate easier preparation and execution of QNLP experiments, we have developed `lambeq`[2,3], an open-source, modular, extensible high-level Python library that provides the tools for implementing a pipeline for experimental QNLP. `lambeq` is under active development, informed by real-world usage.

A high-level overview of a standard QNLP pipeline with `lambeq` is shown below:



`lambeq` provides routines for each of the stages shown, allowing researchers to easily set up a QNLP pipeline for any experiment. Modularity allows any of these stages to be replaced with a custom implementation that is more finely tuned to an experiment's needs, whilst keeping the same high-level structure. Finally, `lambeq` provides a training sub-package that encapsulates the entire pipeline.

In addition to being a library, `lambeq` also provides a web demo[4] and a fully-featured command-line interface, allowing `lambeq` to act as a standalone pregroup [11] parser[5], the first of its kind.

[1] Noisy Intermediate-Scale Quantum

[2] The name is a tribute to mathematician Joachim Lambek (1922-2014), whose seminal work lay at the intersection of mathematics, logic, and linguistics [10].

[3] `https://github.com/CQCL/lambeq`

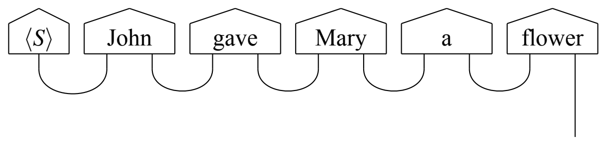
[4] `https://qnlp.cambridgequantum.com/generate.html`

[5] An example of this usage is shown in Appendix B.

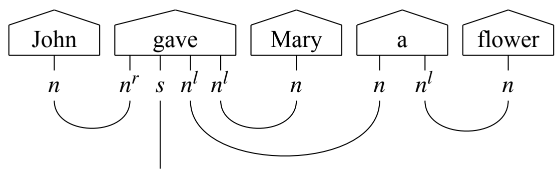
Submitted to:
ACT 2022

2 Parsing and encoding

This first step converts text into a string diagram, based on the graphical calculus for monoidal categories. The underlying data structures for these diagrams are handled by DisCoPy [4], a specialised Python library for manipulating string diagrams. Though DisCoPy is powerful and intuitive to use, it is cumbersome to define large diagrams and the many levels of abstractions which are commonly encountered in QNLP to represent text. Accordingly, `lambeq` defines several ways of easily transforming text into string diagrams. One way is to combine the words in a simple left-to-right manner:



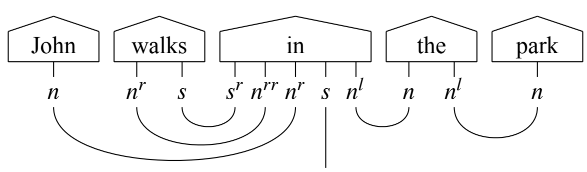
Another way is to follow the syntactic structure of the sentence. A framework for doing this syntax-guided interaction is DisCoCat[6]. The process involves parsing the sentence with a CCG[7] parser, then turning the resulting parse tree into a string diagram [21]:



`lambeq` includes a state-of-the-art CCG parser, named *Bobcat* [1, 2], as well as a simple way to extend the toolkit's functionality to use external CCG parsers, such as *depccg* [22]. Furthermore, `lambeq` also contains a class that provides string diagram conversions for the entire CCGBank corpus [8][8].

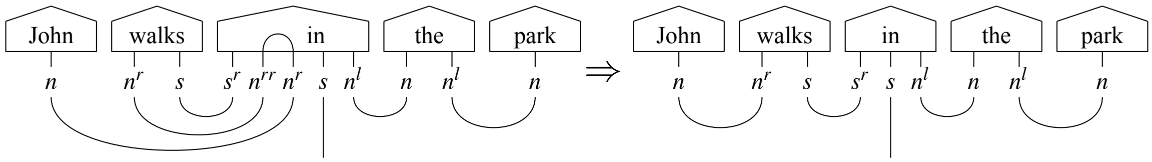
3 Rewriting

Some diagrams can be simplified by incorporating prior assumptions about word interactions. This is helpful for reducing the resource usage of quantum computers when training. `lambeq` includes not only routines for rewriting diagrams, but also a number of standard rewrite rules. For example, the prepositional phrase rewrite rule can be used in the sentence "John walks in the park", the original diagram of which is the following:



This simplification expresses a prior assumption about the preposition "in", namely that the subject noun wires can be bridged by a cap (∩) in the underlying compact-closed monoidal structure, and then "pulled out":

[6] *Dis*tributional *Co*mpositional *Cat*egorical [3].
[7] Combinatory Categorial Grammar [19].
[8] CCGBank consists of 49,000 human-annotated CCG syntax trees, converted from the original Penn Treebank[13]



This reduces the order of the preposition tensor by 2, lowering resource usage in both quantum circuits and classical tensor networks.

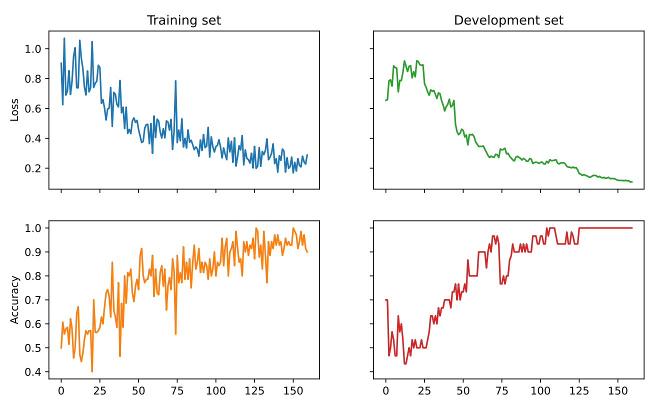
4 Parameterisation

In this module, the symbolic string diagram is turned into a concrete quantum circuit for training, or a tensor network for classical experiments. This is done by applying ansätze, which are maps that determine specific choices in the circuit or network, such as the number of qubits associated with each wire of the string diagram. `lambeq` provides several ways for applying ansätze in quantum and classical cases. For example, `IQPAnsatz` is a class that turns a string diagram into a standard IQP[9] circuit.[10]

5 Optimisation and training

The outputs of the pipeline in `lambeq` can be chosen to work with a number of existing optimisation libraries, including NumPy [6], PyTorch [16], JAX [5] and t|ket⟩ [17], which interfaces with several quantum hardware platforms. Furthermore, `lambeq` provides a training sub-package that can manage the optimisation itself.

To demonstrate its capabilities, we train a model to classify simple sentences from the meaning classification dataset in [12] into two categories: Food or IT. We convert the 130 sentences from the dataset into string diagrams, and then into quantum circuits using `IQPAnsatz`. We optimise using *Simultaneous Perturbation Stochastic Approximation* [18]. We evaluate the model on `qiskit`'s [20] Aer simulator[11], using a noise model to best approximate real quantum hardware. The results show that the model is able to converge to perfect accuracy:



Further details about this experiment, including parameterisation details and other pipelines, are available in [9], as well as further information about the `lambeq` package itself.

[9] Instantaneous Quantum Polynomial: a circuit which interleaves layers of Hadamard gates with diagonal unitaries [7].
[10] An example of a generated IQP circuit is shown in Appendix A.
[11] This is available through t|ket⟩'s Python interface, `pytket`.

References

[1] Stephen Clark (2021): *Something Old, Something New: Grammar-based CCG Parsing with Transformer Models*, doi:10.48550/ARXIV.2109.10044.

[2] Stephen Clark & James R. Curran (2007): *Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models*. *Computational Linguistics* 33(4), pp. 493–552, doi:10.1162/coli.2007.33.4.493.

[3] Bob Coecke, Mehrnoosh Sadrzadeh & Stephen Clark (2010): *Mathematical Foundations for a Compositional Distributional Model of Meaning*. *Linguistic Analysis* 36, pp. 345–384, doi:10.48550/ARXIV.1003.4394.

[4] Giovanni de Felice, Alexis Toumi & Bob Coecke (2021): *DisCoPy: Monoidal Categories in Python*. *Electronic Proceedings in Theoretical Computer Science* 333, pp. 183–197, doi:10.4204/eptcs.333.13.

[5] Roy Frostig, Matthew Johnson & Chris Leary (2018): *Compiling machine learning programs via high-level tracing*. Available at `https://mlsys.org/Conferences/doc/2018/146.pdf`.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke & Travis E. Oliphant (2020): *Array programming with NumPy*. *Nature* 585(7825), pp. 357–362, doi:10.1038/s41586-020-2649-2.

[7] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow & Jay M. Gambetta (2019): *Supervised learning with quantum-enhanced feature spaces*. *Nature* 567(7747), pp. 209–212, doi:10.1038/s41586-019-0980-2.

[8] Julia Hockenmaier & Mark Steedman (2007): *CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank*. *Computational Linguistics* 33(3), p. 355–396, doi:10.1162/coli.2007.33.3.355.

[9] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark & Bob Coecke (2021): *lambeq: An Efficient High-Level Python Library for Quantum NLP*, doi:10.48550/ARXIV.2110.04236.

[10] Joachim Lambek (1958): *The Mathematics of Sentence Structure*. *The American Mathematical Monthly* 65(3), pp. 154–170, doi:10.1080/00029890.1958.11989160. Available at `https://www.cs.cmu.edu/~fp/courses/15816-f16/misc/Lambek58.pdf`.

[11] Joachim Lambek (2008): *From Word to Sentence: a computational algebraic approach to grammar*. Polimetrica sas. Available at `https://www.math.mcgill.ca/barr/lambek/pdffiles/2008lambek.pdf`.

[12] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis & Bob Coecke (2021): *QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer*, doi:10.48550/ARXIV.2102.12846.

[13] Mitchell P. Marcus, Beatrice Santorini & Mary Ann Marcinkiewicz (1993): *Building a Large Annotated Corpus of English: The Penn Treebank*. *Computational Linguistics* 19(2), p. 313–330. Available at `https://aclanthology.org/J93-2004`.

[14] Konstantinos Meichanetzidis, Stefano Gogioso, Giovanni de Felice, Nicolò Chiappori, Alexis Toumi & Bob Coecke (2021): *Quantum Natural Language Processing on Near-Term Quantum Computers*. *Electronic Proceedings in Theoretical Computer Science* 340, pp. 213–229, doi:10.4204/eptcs.340.11.

[15] Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice & Bob Coecke (2020): *Grammar-Aware Question-Answering on Quantum Computers*, doi:10.48550/ARXIV.2012.03756.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai & Soumith Chintala (2019): *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Gar-

nett, editors: *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024–8035, doi:10.48550/ARXIV.1912.01703. Available at `https://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[17] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *t|ket⟩: a retargetable compiler for NISQ devices*. *Quantum Science and Technology* 6(1), p. 014003, doi:10.1088/2058-9565/ab8e92. Available at `https://doi.org/10.1088/2058-9565/ab8e92`.

[18] J.C. Spall (1992): *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*. *IEEE Transactions on Automatic Control* 37(3), pp. 332–341, doi:10.1109/9.119632.

[19] Mark Steedman (2001): *The Syntactic Process*. The MIT Press, doi:10.7551/mitpress/6591.001.0001.

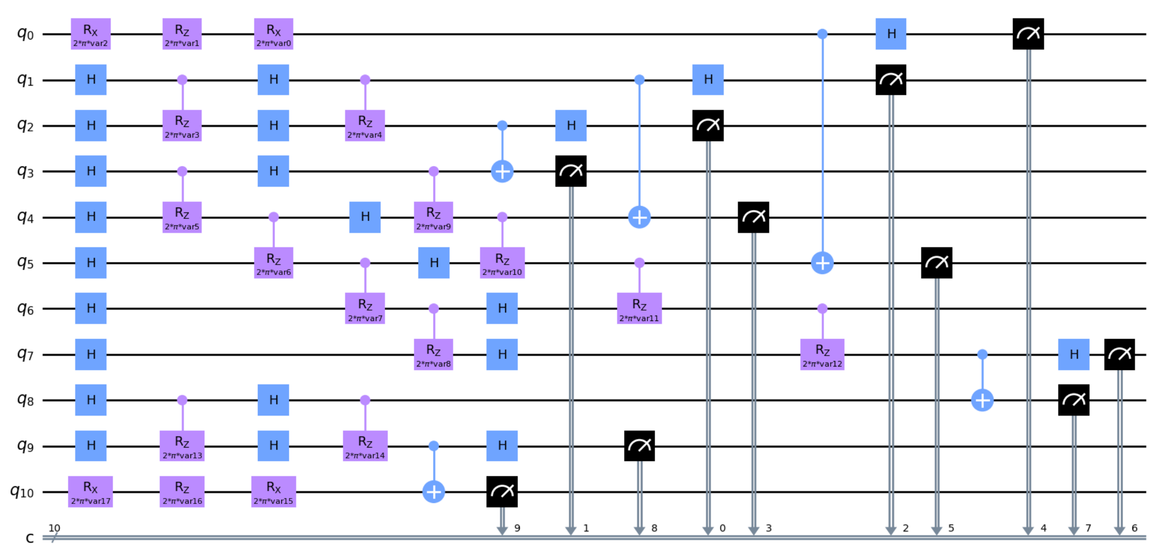
[20] Matthew Treinish, Jay M. Gambetta, SooluThomas, Paul Nation et al. (2021): *Qiskit: An Open-source Framework for Quantum Computing*, doi:10.5281/zenodo.2573505.

[21] Richie Yeung & Dimitri Kartsaklis (2021): *A CCG-Based Version of the DisCoCat Framework*. In: *Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace)*, Association for Computational Linguistics, Groningen, The Netherlands, pp. 20–31, doi:10.48550/ARXIV.2105.07720. Available at `https://aclanthology.org/2021.semspace-1.3`.

[22] Masashi Yoshikawa, Hiroshi Noji & Yuji Matsumoto (2017): *A* CCG Parsing with a Supertag and Dependency Factored Model*. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Vancouver, Canada, pp. 277–287, doi:10.18653/v1/P17-1026.

A IQP Ansatz

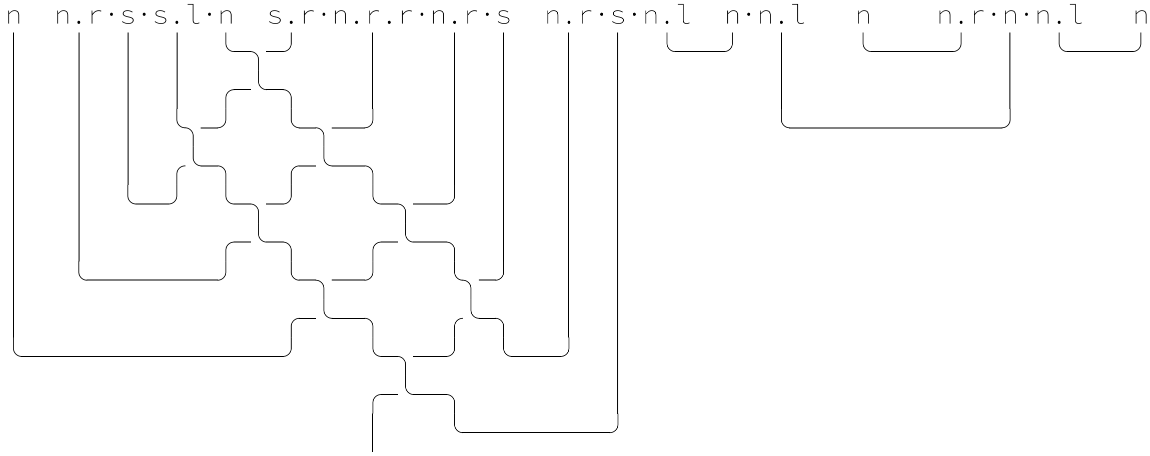
Quantum circuit for the sentence "John walks in the park" in `qiskit` format, where noun and sentence types are assigned 1 qubit each:



B Command-Line Interface

An example of using the `lambeq` command-line interface as a standalone pregroup parser, to parse the sentence "I don't like green eggs and ham":

```
> lambeq --tokenise "I don't like green eggs and ham"
I       do             n't            like    green  eggs     and      ham
─  ─────────────  ───────────────  ─────────  ─────  ────  ─────────  ───
n  n.r·s·s.l·n   s.r·n.r.r·n.r·s  n.r·s·n.l  n·n.l   n    n.r·n·n.l   n
```



The pregroup parser converts the output of a CCG parser into a pregroup derivation [21], which is displayed as a string diagram. The swaps in the diagram correspond to a feature of CCG that is not found in standard pregroup derivations, that of cross-composition.