Automated Derivation of Text Structure as Circuits

Razin A. Shaikh[1,2] Jonathon Liu[1] Benjamin Rodatz[1,3] Richie Yeung[1,2] Bob Coecke[1]
{firstname.lastname}@quantinuum.com
[1]Quantinuum, Oxford
[2]University of Oxford
[3]Technical University of Munich

1 Introduction

DisCoCirc is a novel formalism for language that represents text as circuits [1, 3], and as such, is uniquely well-suited to quantum natural language processing. We implement for the first time an automated process that converts English text to quantum circuits via DisCoCirc, achieving a wide coverage over real-world text. These circuits encode the underlying structure of the text, which includes grammatical information as well as semantic relationships such as coreference and scope.

In the pipeline, we convert syntax trees from categorial grammar into expressions from simply-typed lambda calculus. Then we perform transformations on these lambda terms, introducing semantic information to the original syntactic parse. Finally, we give an abstract string diagrammatic representation to these lambda terms.

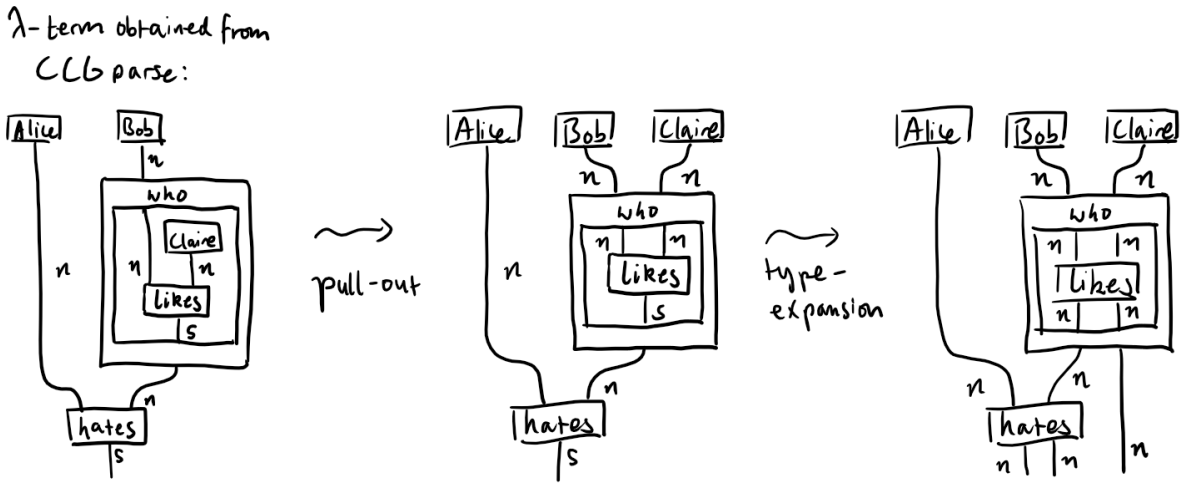
Once the abstract circuit for a text is obtained using the pipeline, one can functorially endow the circuit with semantics in many ways. Of specific interest in view of quantum NLP is to convert the circuits into a parametrized quantum circuit. Indeed there is ongoing work on using the DisCoCirc structures output by this pipeline to solve simple question-and-answer natural language processing tasks, using both classical neural net and PQC approaches. We have implemented this pipeline, and include an automatically generated circuit in the appendix. We also plan to release the pipeline as an open-source package.

2 A simple example

Given a sentence like "`Alice hates Bob who likes Claire.`" we firstly obtain the CCG parse (augmented with some additional grammatical information) using a parser from lambeq [2], a python library for quantum natural language processing. We turn this CCG parse tree into a simply-typed lambda term, which involves collapsing the left and right function types of CCG into a single function type. Given any term of this form (with primitive types n, p, s), we are able to draw it as a circuit-like diagram, consisting of wires, gates that act on wires, and high-order 'frames' that act on gates. We read our diagrams from top to bottom.

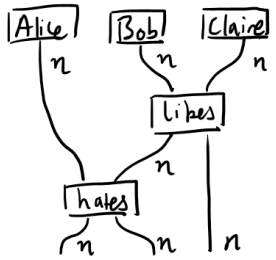
At this point, we perform a number of manipulations to the lambda term. In DisCoCirc we want to compose circuits corresponding to different sentences along their noun wires to build the circuit of the entire text. To enable this, all the noun entities in a diagram must be exposed at the top of the diagram, to allow for precomposition. In this example however, "Claire" is initially trapped inside a frame. We apply a procedure called 'pulling out' to the term which, as the name suggests, corresponds at the diagram level to pulling the noun out of the frame.

Submitted to:
ACT 2023



An additional procedure called 'type expansion' is needed to enable the composition of circuits a la DisCoCirc — the outgoing wires of a diagram must be expanded into some number of n-type wires, where each wire corresponds to one of the entities discussed in the diagram. Only then can we postcompose with other circuits. In previous work on DisCoCirc, there was already some notion of expanding s-type wires into a number of n-type wires corresponding to the constituent entities. Indeed, in this example, it is clear that both occurrences of the s wire should be replaced by two n wires (to preserve the number of n wires going in).

At this stage, our example sentence has been turned into what is essentially a valid DisCoCirc circuit. However, we can additionally choose to apply 'semantic rewrites' that fill in the content of some of the boxes. In this example, we may argue that the "who" frame is semantically devoid of content and interpret it as an identity so that it vanishes.



Note this circuit is exactly what we would obtain from the text "`Bob likes Claire. Alice hates Bob.`" after we compose the two sentence circuits.

References

[1] Bob Coecke (2021): *The mathematics of text structure*. *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pp. 181–217.

[2] 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*. *arXiv preprint arXiv:2110.04236*.

[3] Vincent Wang-Mascianica, Jonathon Liu & Bob Coecke (2023): *Distilling Text into Circuits*. *arXiv preprint arXiv:2301.10595*.