Week 4: Quantum Natural Language Processing

Richie Yeung, Konstantinos Meichanetzidis

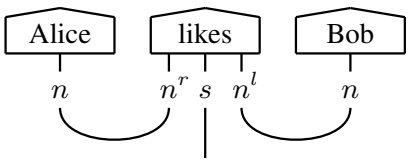
July 2021

Introduction

When you type a search query into your preferred search engine, the search engine does not *explicitly* take grammar into account. In fact, you probably enter your search query as a string of relevant keywords, rather than a question in normal natural language. This is but a simple and prominent use case of natural language processing (NLP). The state of the art in NLP, as with any AI subfield, is claimed by deep learning architectures. However, the inability of current machine learning algorithms to model grammar and context in a *robust way* is one of the reasons they do not generalise well on long complicated sentences, and why machines don't still don't understand language at a human level. To go even deeper, one might say that grammar reflects the structures in the world which we observe and with which we interact. And so, one might then say that a machine would at least need to understand grammar in order to participate meaningfully in the world.

With that in mind, we are motivated to build NLP models that inherently take grammar into account. There exists a rich literature on formal grammar theory and philosophy of language for analysing linguistic structures, of *how words compose* in sentences, and how meaning interacts with context. In parallel, in the last decade, machine learning methods have made great progress by using *vector representations* of the meanings of words. Enter the DisCoCat (Distributional Compositional Categorical) model, introduced 10 years ago [5], which attempted to create vector embeddings for sentence-meanings by composing word vectors according to grammatical structure.

Now, by an algebraic analogy between the mathematics of natural language and of quantum theory, one can choose to use the quantum Hilbert space as the vector space, or "feature space", in which meanings are embedded. The particular "pregroup" grammar model that we use was introduced by Joachim Lambek (for a quick review see [8]). Quantum DisCoCat then invokes categorical quantum mechanics to interpret grammatical reductions, or parsings, as quantum processes that prepare quantum states carrying the meaning of a sentence. Equipped with a pipeline that maps sentences to quantum processes, we are ready to define NLP tasks and tackle them with readily available NISQ processors.



How does DisCoCat work?

First, we begin with a pregroup grammatical derivation of the sentences. This grammatical derivation is then represented as a monoidal diagram (see above). The central idea is that such diagrams represent processes, a la functional programming. Diagrams are read from top to bottom, and they are composed of elementary processes depicted as boxes with input and output wires which carry types. States are boxes with no input, and so they are inputs themselves. Effects are boxes with no output, and so they are tests for outputs. Importantly, when boxes are composed by joining outputs to inputs, only compositions that type-check are allowed.

Coming back to pregroup grammars, a parser assigns some types to words. A word is a state outputing as many wires as the number of its types. Now, types have left- and right-adjoints which dictate between which wires the special CUP effect is to be applied; to put it simply, the types and their adjoints tell the wires of different words how to get joined with each other, leaving only a single dangling wire of sentence type s. Such a sentence diagram is seen as proof of grammaticality of the sentence.

This was the compositional part. In order to endow this diagram with distributional meaning, all we need to do is reinterpret it as a quantum process, where word-states are (pure) quantum states and CUPs are "Bell" (maximally entangled) effects. Of course, every type gets mapped to a vector space dimension, or equivalently a number of qubits. These are seen as hyperparameters that specify the meaning model. Regarding the quantum word embeddings, consider a set of classical parameters for each word, describing the preparation of the state. These parameters could be angles (phases) of an ansatz circuit defined on a number of qubits specified by the types of the word.

Finally, after both the compositional and the distributional parts of the setup have been specified, one needs to just run this quantum process by preparing the quantum word-states, and then apply the "grammar-aware" Bell effects. Thus, one has prepared a quantum state which encodes the meaning of the whole sentence in a Hilbert space of dimension 2^{q_s}.

Recommended Reading

For a high-level informal exposition of quantum DisCoCat with focus on NISQ devices, see [4]. Specifically, for sentence classification experiments, see [10, 9], the former of which was written with a physicist audience in mind and the latter one was targeted at the NLP engineer and practitioner.

Suggested Weekly Tasks

The following tasks will guide you towards running your own natural language experiments on a quantum computer. This tutorial assumes that you have experience with programming in Python.

Task 1: Learning how to use DisCoPy

Read and complete the exercises from this notebook [1]. This notebook will show us how to use DisCoPy, the library for representing and manipulating DisCoCat diagrams. In particular, after learning how to represent cooking recipes within the framework of monoidal categories, we will learn how to use and construct our own functors. Functors are a structure preserving map that enables us to convert DisCoCat diagrams to tensor networks and quantum circuits with ease.

Task 2: Generating your own dataset

Since the work of [9], there has been a paper [14] which utilises CCG grammars instead of pregroup grammars to construct DisCoCat diagrams. Due to the ubiquity of CCG parsers, we now have the means to construct diagrams from unparsed sentences. A web demo of this tool is available at

`https://qnlp.cambridgequantum.com/generate.html`

And there is a direct API available here:

Parse Tree	(png)	(code)
DisCoCat Diagram	(png)	(code)

Alternatively you can generate sentences using your own fixed templates, based on the POS (Parts Of Speech) tags of the words. This is preferable for short sentences as the results of the parser are merely statistical.

For this task, decide what NLP task you would like to tackle, then generate your own dataset for that task. The notebook we provide is designed for binary classification (e.g. document classification, spam detection, sentiment analysis) but you may want to come up with another task.

Due to time constraints, we suggest you use a corpus of no more than 100 sentences. Start with short sentences of at most 5 words, then incrementally scale up your experiment. Remember the quality of the experiment's results depend a lot on the quality of the corpus. Think and discuss what your task should be and decide on a consistent methodology with which you generate the dataset. Before running the experiment, it may be a good idea to estimate how long your experiment would take to complete.

Task 3: Run your first QNLP experiment

Now that you have a dataset and its DisCoCat diagrams, you are ready to perform your first QNLP experiment. Here is a notebook which contains a small-scale QNLP experiment to get you started.

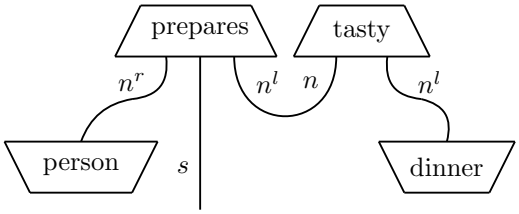
Useful Tip: you may want to use DisCoPy's native `eval` method in the beginning, which is faster than simulating on qiskit's AerBackend. Once you have a working experiment, then you can use the AerBackend to import a noise model to simulate a particular device of your choosing.

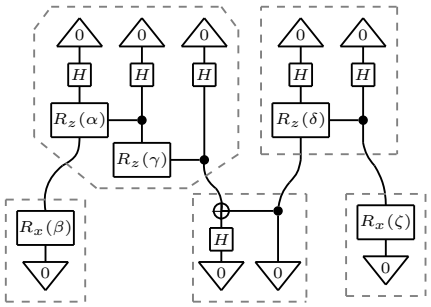
Possible Extensions

Here are some possible extensions to improve the QNLP experiment.

Diagrammatic Rewriting

Before converting the DisCoCat diagrams into circuits, it is possible to first perform some intermediate rewriting. Certain words in the English language have a special role, hence we give them concrete non-parameterised forms made of cups and spiders. See [3] and chapter 5 of [7] for some examples. Another form of rewriting is direct rearrangement: thanks to the work on categorical quantum mechanics [2, 6], we know the meaning of a diagram is preserved under deformation. For example, [9] bends the outside nouns of a sentence to reduce the amount of qubits and postselection used.





Different Ansatz

Try converting the word states into different parameterised ansatz and see how it af fects performance. There is a wide selection of ansatz available in the quantum machine learning literature (See section 4 of [13] for example). Certain words exhibit special linguistic properties, so you might consider using different ansatz for different words, within the same experiment.

Different Optimisers

In the small-scale notebook we use the SPSA algorithm to minimise our loss function. One can try other gradient-free optimisers, too. However, having an analytic gradient often leads to better optimisation performance and accuracy. Using diagrammatic differentiation [12], gradient-based machine learning by using gradient recipes [11] in a compositional fashion. Using the `grad` method built into DisCoPy, write your own optimiser to perform a gradient-based QNLP experiment. How does the gradient-based optimised approach compare with the black-box SPSA optimisation approach, in terms of the amount of quantum computing resources used?

Different grammar model

Other than pregroup; try other parsings. Define other quantum functors: maybe in terms of channels. In other words, try other baselines. Bonus question: try to find a task where you can see that having a model that is grammar-informed does better than a bag-of-words unstructured model.

References

[1] discopy: Qnlp tutorial. `https://discopy.readthedocs.io/en/main/notebooks/qnlp-tutorial.html`. Accessed: 2021-06-30.

[2] Samson Abramsky and Bob Coecke. Categorical quantum mechanics, 2008.

[3] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. The frobenius anatomy of relative pronouns. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 41–51, 2013.

[4] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi. Foundations for near-term quantum natural language processing, 2020.

[5] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning, 2010.

[6] Chris Heunen and Jamie Vicary. *Categories for Quantum Theory: an introduction.* Oxford University Press, USA, 2019.

[7] Dimitri Kartsaklis. Compositional distributional semantics with compact closed categories and frobenius algebras. *arXiv preprint arXiv:1505.00138*, 2015.

[8] Joachim Lambek. Pregroups and NLP. `https://www.math.mcgill.ca/rags/JAC/124/Lambek-Pregroups-s.pdf`.

[9] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis, and Bob Coecke. QNLP in practice: Running compositional models of meaning on a quantum computer. *CoRR*, abs/2102.12846, 2021.

[10] Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, and Bob Coecke. Grammar-aware question-answering on quantum computers, 2020.

[11] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[12] Alexis Toumi, Richie Yeung, and Giovanni de Felice. Diagrammatic differentiation for quantum machine learning. *arXiv preprint arXiv:2103.07960*, 2021.

[13] Richie Yeung. Diagrammatic design and study of ans\"{a} tze for quantum machine learning. *arXiv preprint arXiv:2011.11073*, 2020.

[14] Richie Yeung and Dimitri Kartsaklis. A ccg-based version of the discocat framework. *CoRR*, abs/2105.07720, 2021.