IQ01 - Informatique Quantique

Traitement automatique du langage quantique

Gustave Cortal

Automne 2021 - Walter Schon

1 Traitement automatique du langage quantique

Le langage naturel peut être interprété comme des processus quantiques qui sont représentés sous forme de diagramme. Ce raisonnement par diagramme s'opère grâce au formalisme de la mécanique quantique catégorielle [1] qui permet d'encoder structure et sens linguistique. À partir de cette représentation par diagramme, le ZX-calcul est utilisé pour obtenir des circuits quantiques qui peuvent ensuite être implémentés sur des machines quantiques existantes. Cette section explique comment s'opère le passage du langage naturel aux circuits quantiques en s'appuyant sur les concepts introduits par l'étude *Foundations for Near-Term Quantum Natural Language Processing* [1].

Nous verrons que l'approche quantique permet d'encoder le sens à travers le contexte, en utilisant la structure linguistique des mots. Plus précisément, nous présenterons le modèle quantique pour le traitement du langage naturel, puis nous comparerons l'approche classique avec l'approche quantique. Pour finir, nous réaliserons une démonstration technique en utilisant la nouvelle bibliothèque Python lambeq de Cambridge Quantum [2], qui permet d'effectuer du traitement automatique du langage quantique.

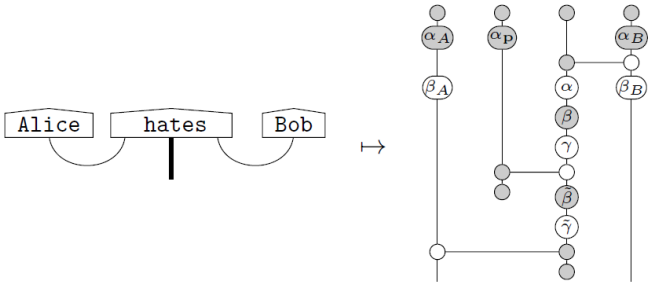


FIGURE 1 – Représentation de la phrase *Alices hates Bob* dans des circuits quantiques

1. La mécanique quantique catégorielle utilise des notions venant de la théorie des catégories pour représenter plus fidèlement les processus quantiques.

1.1 Modèle quantique pour le langage naturel

Nous présentons le modèle quantique pour le traitement du langage naturel à travers deux cas simples : l'application d'un adjectif à un nom et la mise en place d'un sujet et d'un objet dans un verbe pour créer une phrase.

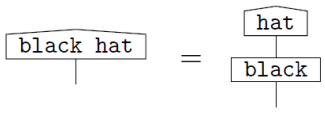
Les sens des mots peuvent être décrits comme des états de qubits. Par exemple, le sens du nom (n) *hat* est l'état d'un qubit $|\psi_n\rangle \in \mathbb{C}^2$.

1.1.1 Application d'un adjectif à un nom

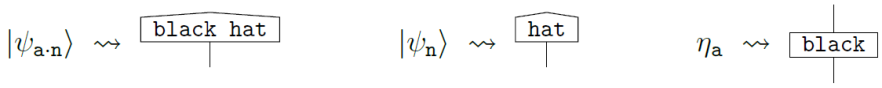
Un adjectif a modifie un nom n. Par exemple, l'adjectif *black* dans *black hat* modifie le nom *hat* pour avoir la propriété d'être noir. L'adjectif a est représenté comme une application linéaire d'un qubit vers un autre soit $\eta_a : \mathbb{C}^2 \to \mathbb{C}^2$. Ainsi, le sens de l'adjectif appliqué au nom est :

$$|\psi_{a.n}\rangle = \eta_a(|\psi_n\rangle) \tag{1}$$

La représentation par diagramme peut également être utilisée, comme illustré ci-dessous :



En utilisant les substitutions suivantes :



Une autre façon d'obtenir le sens de *black hat* est de considérer l'adjectif *black* comme l'état de deux qubits, soit $|\psi_a\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$. Ainsi, l'état d'un adjectif appliqué à un nom peut être écrit de cette façon en notation de Dirac :

$$|\psi_{a.n}\rangle = (I \otimes \langle Bell|) \cdot (|\psi_a\rangle \otimes |\psi_n\rangle) \tag{2}$$

où $\langle Bell| = \langle 00| + \langle 11|$ et I est l'identité.

Cette équation est représentée par le diagramme suivant :

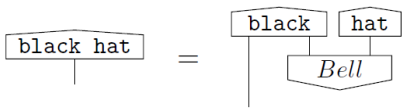
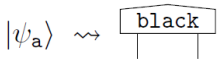


FIGURE 2 – Intrication du sens de *black* et *hat*

Avec la substitution suivante :



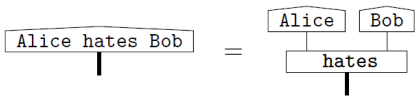
L'avantage de cette forme est de mettre ensemble le sens de *black* et de *hat* en utilisant le produit tensoriel $|\psi_a\rangle \otimes |\psi_n\rangle$. Puis, avec l'application $I \otimes \langle Bell|$, les sens des mots interagissent pour produire un sens global supérieur aux sens des parties. L'application représente donc la grammaire du tout.

1.1.2 Mise en place d'un sujet et d'un objet dans un verbe

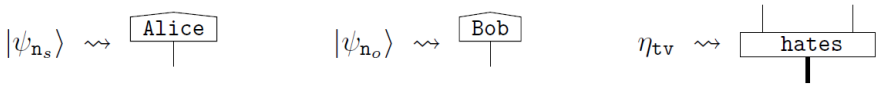
Pour produire une phrase minimale, il faut un verbe transitif tv, un sujet n_s et un objet n_o. Par exemple, le verbe *hate* a besoin d'un sujet *Alice* et d'un objet *Bob* pour produire la phrase *Alice hates Bob*. Le verbe transitif peut être vu comme une application η_{tv} qui prend deux noms $|\psi_{n_s}\rangle \in \mathbb{C}^2$ et $|\psi_{n_o}\rangle \in \mathbb{C}^2$ et produit une phrase :

$$|\psi_{n_s.tv.n_o}\rangle = \eta_{tv}(|\psi_{n_s}\rangle \otimes |\psi_{n_o}\rangle) \in \mathbb{C}^{2k} \tag{3}$$

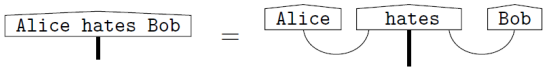
On obtient ainsi le diagramme suivant :



Avec les substitutions suivantes :



En effectuant le même raisonnement que précédemment avec l'adjectif, le verbe transitif peut être représenté non plus comme une application, mais comme un état $|\psi_{tv}\rangle \in \mathbb{C}^2 \otimes (\mathbb{C}^{2k}) \otimes \mathbb{C}^2$. Le sens d'une phrase est donc représenté par ce diagramme :



En revenant à la notation de Dirac, nous avons :

$$|\psi_{n_s.tv.n_o}\rangle = (\langle Bell| \otimes I \otimes \langle Bell|) \cdot (|\psi_{n_s}\rangle \otimes |\psi_{tv}\rangle \otimes |\psi_{n_o}\rangle) \tag{4}$$

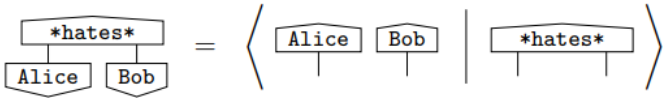
Ainsi, comme précédemment, le produit tensoriel entre le sens des mots $|\psi_{n_s}\rangle \otimes |\psi_{tv}\rangle \otimes |\psi_{n_o}\rangle$ est effectué, puis à travers l'application $\langle Bell| \otimes I \otimes \langle Bell|$, la grammaire est appliquée pour faire interagir les différents sens et produire le sens global de la phrase.

Les équations sont obtenues à travers un algorithme général, nommé DisCoCat [3], qui produit le sens d'un tout à travers les mots qui le constituent. Tout d'abord, il s'agit d'effectuer

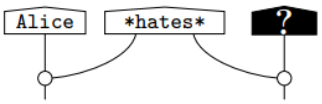
le produit tensoriel entre les états des différents mots qui constituent le tout. À ce stade, tous les mots ne sont pas intriqués, il n'y a donc aucune interaction. C'est la structure grammaticale qui va faire interagir les mots entre eux pour produire un tout intriqué, en utilisant uniquement des effets de Bell et des identités.

Grâce au ZX-calcul, les diagrammes obtenus avec l'algorithme DisCoCat sont ensuite convertis en circuits quantiques utilisant seulement des portes quantiques unitaires à un qubit et des portes CNOT. C'est donc un langage universel puisqu'à partir d'une porte CNOT et d'une porte unitaire à un qubit arbitraire, tous les circuits peuvent être générés. Il est également montré que le ZX-calcul génère n'importe quelle application linéaire, ce qui fait de ce calcul un langage graphique puissant permettant de raisonner par rapport aux applications linéaires entre les qubits à l'aide de diagramme. Nous avons maintenant toutes les clés en main pour effectuer certaines tâches très utilisées dans le traitement automatique du langage.

1.1.3 Tâches de traitement automatique du langage



Il y a un lien entre le degré de vérité d'une phrase et son sens. Ainsi, le produit scalaire mesure à quel point la paire Alice et Bob correspond à la relation *hates*. Grâce à cela, la recherche du vecteur le plus proche peut facilement être implémentée pour un problème donné. Par exemple, si nous voulons trouver le sujet qui maximise le degré de vérité de la phrase *? hates Bob*, il suffit de calculer plusieurs fois le produit scalaire en prenant les potentiels candidats, et de choisir le candidat qui maximise la relation :



Le même raisonnement est effectué pour n'importe quel mot dans une phrase. Par exemple, nous pouvons également chercher le verbe qui maximise la relation entre Alice et Bob ou le sujet et l'objet qui se détestent le plus :

Pour calculer la similarité entre deux phrases, il suffit de calculer le produit scalaire entre le sens global de chaque phrase. Il est important de noter que nous pouvons seulement comparer des entités linguistiques de même type puisqu'il faut que les dimensions coïncident. Par exemple,

il n'est pas possible de comparer un nom et un adjectif. Dans la prochaine partie, les différences majeures entre l'approche classique et l'approche quantique sont expliquées.

1.2 Pourquoi utiliser l'approche quantique ?

— **Une prise en compte du devenir du sens.** La représentation par diagramme permet d'illustrer le flux de sens entre les mots. En appliquant la grammaire, le sens du sujet et de l'objet arrive dans le verbe. Considérons un texte quelconque, si un nom apparait dans plusieurs phrases différentes, ces dernières vont devenir intriquées. Les sens ne sont donc plus statiques, comme dans le traitement du langage naturel classique, mais sont en devenir puisqu'ils évoluent au fur et à mesure que le texte avance. Ainsi, une phrase doit être vue comme un processus qui altère le sens de ses mots.

— **Réduction de la charge computationnelle.** L'utilisation de la représentation vectorielle pour encoder les mots est commune entre l'approche classique et l'approche quantique du traitement automatique du langage. Cependant, cette dernière possède un avantage pour stocker les représentations vectorielles. En effet, un système à n qubits ayant 2^n degrés de liberté, il est possible d'implémenter un mot à N dimensions avec seulement $log_2(N)$ qubits. Par exemple, pour stocker un vecteur à 2000 dimensions, il faut 8×10^9 bits pour un ordinateur classique et 33 qubits avec le quantique [4]. Si nous passons à 10 000 vecteurs à 2000 dimensions, il faudra 8×10^{13} bits contre seulement 47 qubits.Notons que dans l'approche classique, le sens des mots est encodé dans des vecteurs de même dimension alors qu'avec l'approche quantique, chaque mot possède sa propre dimension selon son type grammatical.

— **L'introduction d'une nouvelle logique.** Le langage naturel et le quantique partage une logique d'interaction, ce qui fait qu'il est intéressant d'avoir une approche quantique pour modéliser le langage. En effet, l'approche quantique nous incite à penser l'interaction entre les systèmes, en évitant le réductionnisme. Le sens d'une phrase ne peut être réduit aux mots qui la constituent. Il faut prendre en compte comment les mots interagissent entre eux pour produire du sens.

— **La possibilité de représenter l'ambiguïté des mots.** On peut également utiliser des matrices de densité, et non plus des vecteurs d'états, pour représenter le sens des mots. L'ambiguïté du sens d'un mot peut ainsi être représentée par l'état mixte d'une matrice de densité. Chaque mot est donc perçu comme un mélange probabiliste des différents sens qu'il peut avoir. Le mot *reine* peut représenter une abeille, une pièce d'échec ou un personnage royal. Pour prendre en compte ces différents sens, l'état mixte suivant peut être formé : $\rho_{reine} = |reine - abeille\rangle \langle reine - abeille| + |reine - echec\rangle \langle reine - echec| + |reine - royal\rangle \langle reine - royal|$

— **Le sens réside dans le contexte**. L'approche quantique pour modéliser le langage naturel résonne avec certaines traditions en philosophie du langage, notamment le holisme sémantique. Par exemple, Frege et Wittgenstein sont partisans du « principe de contexte »qui stipule que le sens d'un mot émerge du contexte seulement, on ne peut donc pas isoler un mot pour sa compréhension [5]. De plus, le raisonnement par dia-

gramme, qui représente la grammaire par des câbles entre les mots, est analogue à la pensée de Chomsky [6] qui présentait la grammaire universelle comme une structure profonde encodée dans nos circuits cérébraux.

En conclusion, l'approche quantique pour le traitement du langage naturel est plus qu'une alternative à l'approche classique. Nous avons voulu mettre l'accent sur le fait que le quantique nous donne une opportunité de modéliser le langage d'une nouvelle façon. La logique d'interaction et l'encodage du sens et de la structure linguistique dans des circuits quantiques nous permet de dépasser les boites noires de l'apprentissage machine moderne pour créer des systèmes se concentrant sur les flux de sens entre les mots grâce à la grammaire. Dans la prochaine partie, nous présenterons comment une phrase est représentée sous la forme d'un circuit quantique à l'aide de la bibliothèque lambeq.

1.3 Exemple avec la bibliothèque lambeq

Cette section repose sur le papier très récent *lambeq : An Efficient High-Level Python Library for Quantum NLP* [2] paru en octobre 2021. La bibliothèque et sa documentation sont disponibles sur Github [2]. Nous présenterons le passage d'une phrase à un circuit quantique à travers un exemple simple.

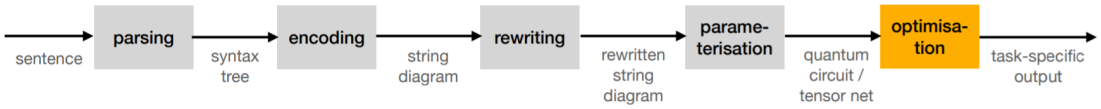


FIGURE 3 – *Pipeline* implémenté dans lambeq

La première étape est d'identifier les parties du discours d'une phrase (*parsing*) pour ensuite dériver son arbre syntaxique. Cela consiste à assigner pour chaque mot son type grammatical (nom, adjectif, verbe, etc.). Ensuite, un diagramme est généré grâce au formalisme de la grammaire du prégroupe. Un prégroupe est un affaiblissement de la notion de groupe où il existe une distinction entre l'inverse à gauche et l'inverse à droite. De plus, l'égalité est remplacée par un ordre. Les relations suivantes sont obtenues : $x^l \cdot x \leq 1 \leq x \cdot x^l$ et $x \cdot x^r \leq 1 \leq x^r \cdot x$ où x^l et x^r sont respectivement les adjoints à gauche et à droite de x.

Pour la phrase *Gustave loves quantum*, le diagramme suivant est obtenu :

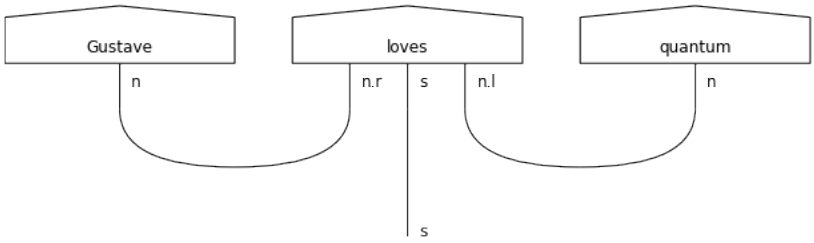


FIGURE 4 – Représentation par diagramme de *Gustave loves quantum*

Le type grammatical de *loves* est $n^r \cdot s \cdot n^l$. Cela signifie que le verbe attend un nom à gauche et à droite pour produire la phrase s. Une phrase est grammaticalement correcte si son type grammatical est inférieur à la phrase s. La vérification s'effectue à l'aide des relations définies ci-dessus : $n \cdot n^r \cdot s \cdot n^l \cdot n \leq s \cdot n^l \cdot n \leq s$. Nous voyons que nous disposons d'un formalisme puissant pour représenter une phrase comme un diagramme, indépendamment des décisions à plus bas niveau, comme le choix des portes quantiques. Cette forme abstraite peut être représentée par des circuits quantiques concrets en appliquant un *ansatz*. Ce dernier peut être vu comme une application qui détermine des choix comme le nombre de qubits à assigner à chaque câble du diagramme ainsi que la paramétrisation des états quantiques qui correspondent à chaque mot.

2. https ://github.com/CQCL/lambeq

Nous appliquons l'*ansatz* IQPAnsatz en assignant un qubit pour chaque nom et pour chaque phrase. Le circuit *qiskit* suivant est obtenu :

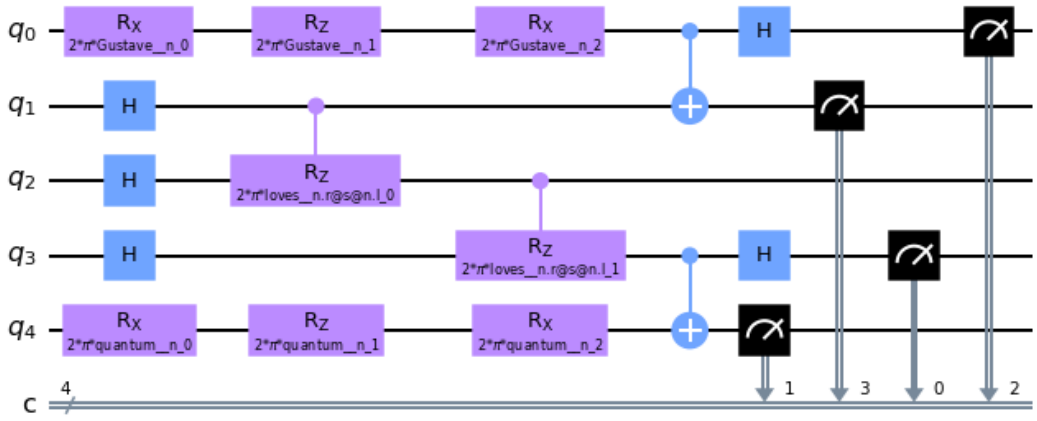


FIGURE 5 – Circuit quantique implémenté avec la bibliothèque *qiskit* correspondant à la phrase *Gustave loves quantum*

Notre phrase peut maintenant être implémentée sur n'importe quelle machine quantique et être utilisée pour l'entraînement d'un modèle d'apprentissage automatique. Le code utilisé pour produire ce circuit est disponible avec le rapport sous le nom « lambeq.ipynb ».

Références

[1] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi. *Foundations for Near-Term Quantum Natural Language Processing*, 2020.

[2] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. *lambeq : An Efficient High-Level Python Library for Quantum NLP*, 2021.

[3] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. *A compositional distributional model of meaning.* In *Proceedings of the Second Symposium on Quantum Interaction*, 2008.

[4] William Zeng and Bob Coecke. *Quantum Algorithms for Compositional Natural Language Processing. Electronic Proceedings in Theoretical Computer Science*, 221 :67–75, Aug 2016.

[5] Gottlob Frege. *The Foundations of Arithmetic : A Logico-Mathematical Enquiry Into the Concept of Number.* New York, NY, USA : Northwestern University Press, 1950.

[6] Noam Chomsky. *Aspects of the Theory of Syntax.* The MIT Press, Cambridge, 1965.