QHACK 2021 : Variational Language Model

Slimane Thabet, Jonas Landman [TEAM X]

February 26, 2021

1 Introduction

For the QHACK hackathon, we[1] have imagined a new variational quantum algorithm for Natural Language Processing (NLP). Our goal is to train a quantum circuit such that it can process and recognize words. Our work includes:

- An original word encoding on qubit using deep learning pre-processing for word embeddings. It allows a sentence of several words to be the input of a quantum circuit.
- A custom ansatz with alternating layers of word Shuffling and word Entanglement, meant to process the sentence.
- A wrapper circuit that uses SWAP Test to compare one word in the sentence and a given supplementary word.
- A training strategy for the circuit learn how to recognize a missing word in a sentence.

Applications varies from word matching, sentence completion, sentence generation named entity recognition and more.

The latent Hilbert space grow exponentially in the number of qubits. If we encode a word on 3 qubits, a sentence of 10 words will live in a $2^{30} > 10^9$ latent space. Computing the overlap of two sentences in such a configuration could give us a similarity metric very hard to estimate with a classical computer. The relation between the words can be better taken into account to construcrt powerful sentence and word embeddings. In [1] the authors develop this idea by creating quantum states of sentences from the grammatical relationship between words. Our method is in the same spirit, but we don't use the grammatical relationships. We inspire ourselves from the transformers architecture to derive a contextual based word embedding [2]. Grammatical relationships are indeed very hard to get, and our method has the advantage to generalize to any number of words or sentences.

[1] This project is the first collaboration between the authors

2 Word encoding

As usual in NLP, words are represented by vectors. We chose to preprocess words using state-of-the art deep learning word embedding methods such as FastText [3]. Then these embeddings are cast down to few features using dimensionality reduction techniques such as PCA. For instance each word will be described as a vector of 8 dimensions. Using Quantum Amplitude Encoding , we can encode each word into a 3-qubits register. If a sentence is composed of N words, and to represent it we propose to stack N 3-qubits register sequentially.

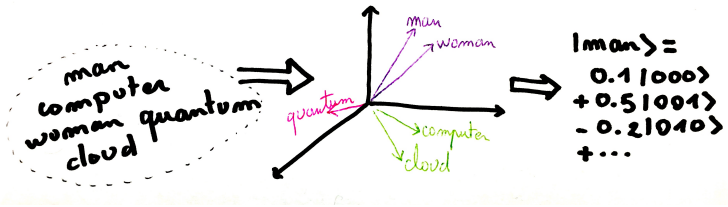

Figure 1: From words (left) to word embeddings (middle) to quantum words (right)

3 Variational Circuit

We propose a new ansatz and training methodology to perform this NLP quantum learning:

- The ansatz (Fig.3) is composed of several layers of controlled rotations that mix the words between each other (Fig.4), and between themselves (Fig.5).
- During the training, we will mask one word randomly in each sentence, by imposing its quantum register to $|0\rangle$
- Using a SWAP Test, a supplementary word is then compared to the output register of the missing word (after the output of the ansatz). Therefore the cost function is the probability of measuring '0' on the swap test's ancillary qubit. We chose the supplementary word to be the missing word itself in order to drive the learning. See Fig.2.
- The goal of the training is to adjust the ansatz's parameters such that the missing word is guessed.

SWAP Test: The SWAP Test consists in applying several controlled-SWAP gate on two quantum states $|\psi(\theta)\rangle$ and $|\phi\rangle$, controlled by an ancillary qubit. Here $|\psi(\theta)\rangle$ is one output of our adjustable ansatz, and $|\phi\rangle$ is the desired answer. Before and after the SWAP gates, we apply an Hadamard gate on the ancillary qubit. The probability of measuring '0' on the ancillary qubit is proportional to the inner product or *overlapping* $\langle\psi(\theta)|\phi\rangle$ that we want to maximize.

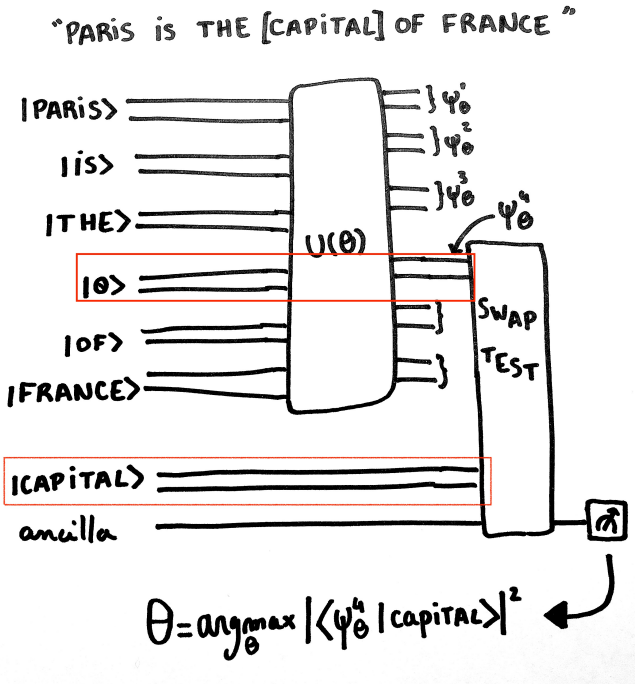

Figure 2: Final circuit

The Ansatz itself is constructed with layers of alternating structures. Each layer is composed of one sublayer of *Shuffling* (Fig.4) and one sublayer of *Entanglement* (Fig.5). Both sublayers are made of controlled-RY gates. The *Shuffling* is applied independently on each word, with the same set of parameters each.

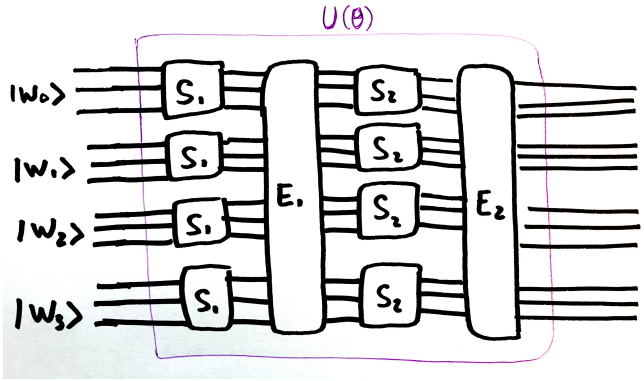

Figure 3: Details of our alternating layer ansatz

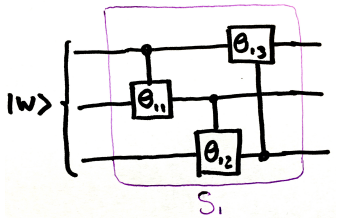

Figure 4: Shuffling sublayer (S)

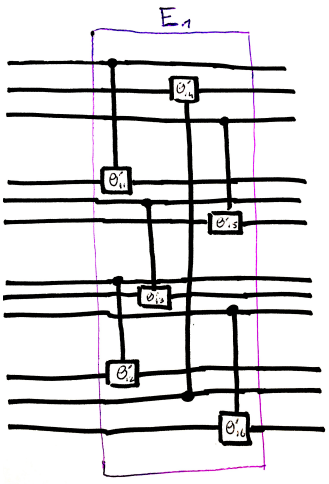

Figure 5: Entanglement sublayer (E)

4 Applications

4.1 Missing word retreiving

With such a circuit trained, the main application is to provide a new sentence with a missing word and compare it with all possible words in the "dictionary". Due to the training methodology, our circuit is trained to recover the missing word.

4.2 Sentence generation

Another application is to generate artifical sentence by starting with only one word, or completing a sentence after its last words. This is simple application due to the way the circuit has been trained in our setting.

4.3 Named Entity Recognition

A third application is called *Entity Recognition*, where one wants the algorithm to classify words into categories (e.g. places, people). In our case, it implies to train a second ansatz called the decoder. The decoder is applied after the encoder at the position we want to get the category, and the result of the measurement gives us the category of the word.

5 Resources

We consider M sentences of N words, each one encoded as Q qubits.

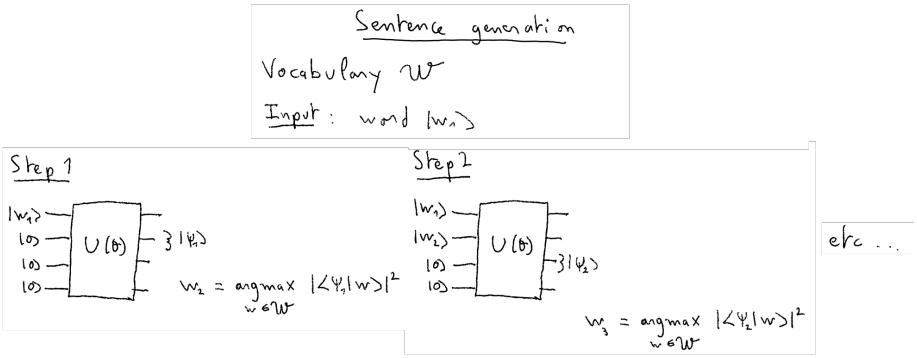

Figure 6: An application : Sentence Generation

- Number of qubits required : One quantum circuit corresponds to one sentence plus an extra word and an ancillary qubit, therefore $Q*(N+1)+1$ qubits. E.g for a 4 words sentence with 3 qubits per words, we require 16 qubits. For a 5 words sentence with 4 qubits per words, we require 25 qubits.
- Number of trainable parameters : The number of trainable parameters in the ansatz is around $Q*(1+N/2)*L$, where L is the number of layers, on average (it depends of the parity of the number of words, and number of qubits). E.g for a 4 words sentence with 3 qubits per words and 3 layers, we require 27 parameters.

We used the AWS SV1 simulator for parallelizing the gradient computations during the training. But the computational cost remains high due to the number of sentences and the total number of words in the dictionary.

6 Datasets

We propose 3 differents datasets to train and test our algorithm

- IMDB Dataset composed of 10k sentences and 12 words in total
- Newsgroup Dataset composed of 10k sentences and 12 words in total
- An synthetic dataset of 'dummy' sentences with small number of sentences and words, for performance limitation and grammatical simplicity. For instance a 5 words sentence follows the structure **adjective + name + verb + adjective + name** taken at random in a custom list a words for each category.

7 Results

Thanks to the Power Up credits provided by AWS Braket, we were able to train our algorithm during several hours on the SV1 simulator. We started with our

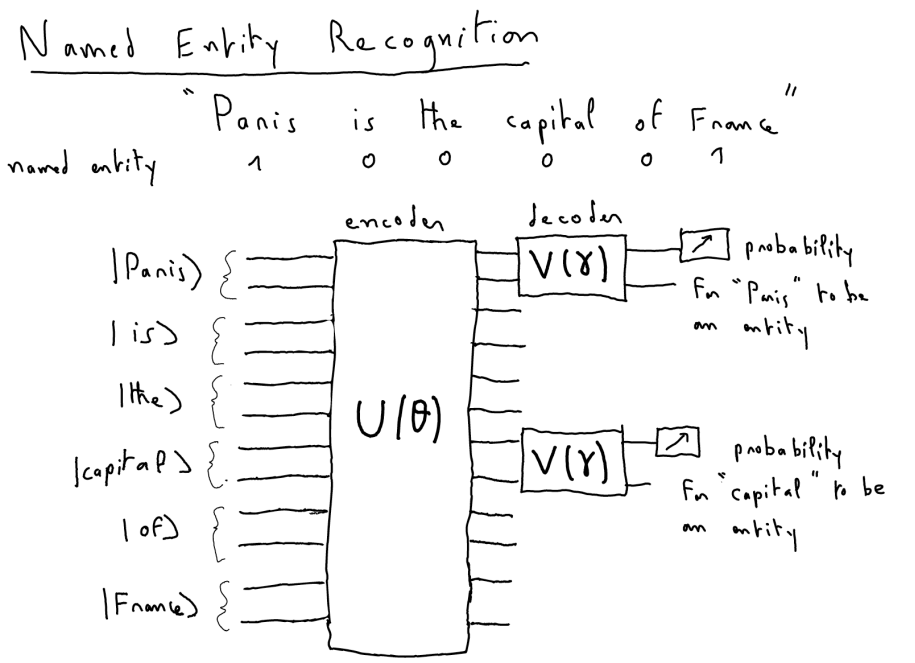

Figure 7: An application : Named Entity Recognition

synthetic dataset that allowed a smaller number of words in the dictionary, and sentences of small size.

Results showed that the loss associated to our ansatz and training methodology was decreasing during training. See Fig.8. One can see that the convergence has not been reached yet, hence a longer training would probably yield even better results. The circuit considered here used a two layers ansatz and had only sentences of 5 words, each encoded in 2 qubits. Therefore we simulated 13 qubits and 30 parameters. The training was stopped after 12 hours and 25k sentences.

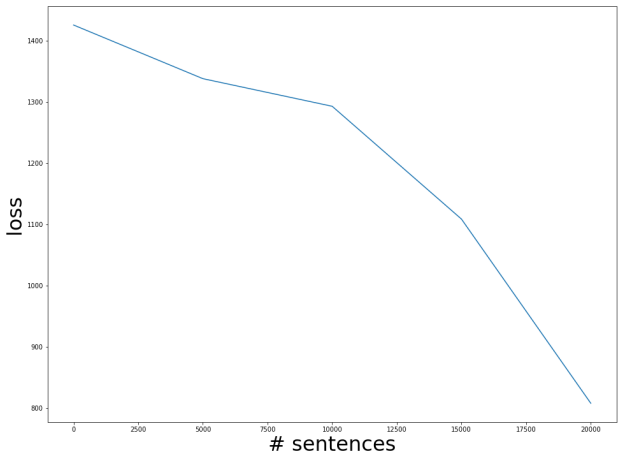

Figure 8: Traingin loss on our synthetic dataset during 12h with AWS SV1 simulator. 25k sentences (5k per epoch) each of 5 words, with 2 qubits per word and a 2 layers ansatz.

As a simple test application, we asked our circuit to complete the following sentence **"funny $\cdots$ eat cheap vegetable"**. The two most probable words given by our circuit were **bird** and **cat**. For the *sentence generation* task, we obtained results such as **"kid teacher policeman old man"** which does not really make sense as a sentence, but we can see that the circuit make good association between semantically close concepts! In the *named entity recognition* task, we tried to make our circuit recognize verbs, but it was mostly a failure, with an accuracy of classification of less than 50%.

Although this certainly doesn't constitute a proof that our quantum circuit have learned how to read, it is a simple a encouraging example for this hackathon.

Since many more gradient descent iterations are required, some other tests were less conclusive. For instance we have noticed that the circuit has usually trouble with outputing the right *verb* (middle of the sentence). We can also see the limitation of our synthetic dataset were sentences are created at random, which implies many absurd example.

We conducted additional experiments, on different dataset, with sometimes more layers on the ansatz, more words and more qubits per words. However the time available to train these configurations was certainly not enough to obtain interesting results. See Fig.9.

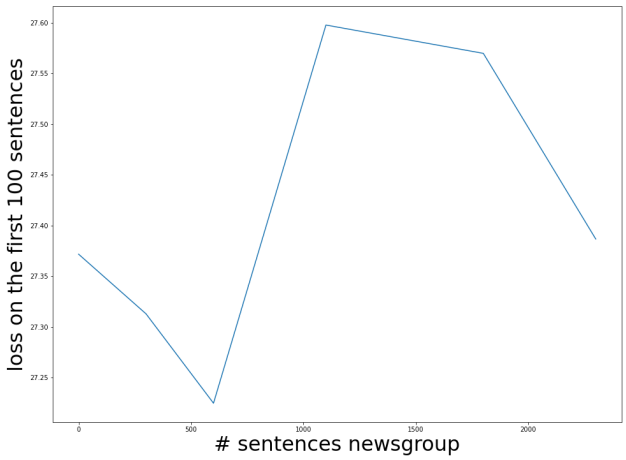

Figure 9: Training loss on the Newsgroup dataset during 10h with AWS SV1 simulator. 20k sentences (5k per epoch) each of 10 words, with 2 qubits per word and a 2 layers ansatz.

8 Code architecture

- The Pennylane variational ansatz are defined in *utils.py*
- The NLP preprocessing using FastText is made in *preprocessing_dataset.py* and generate readable file as *embeddings.npy*, *sentences.npy* etc.
- In *config.py* are defined the global configurations such as the number of words, of qubit per words, and the number of layers per ansatz.
- In the notebook *Final_notebook*, we train the quantum variational circuit and test applications

9 Conclusion and Perspectives

In this hackathon project, we have developped a new circuit design for Natural Language for which we obtained encouraging results. To our knowledge, this is the first time such an approach is proposed with a quantum variational quantum circuit for NLP applications.

There are many more ideas to try in order to improve the efficiency of our circuit. Here is a non-exhaustive list:

- Change the wrapper circuit and include the supplementary word inside the ansatz as well, as shown in Fig.10.

- Try a circuit with a deeper ansatz, because so far the number of parameters is rather low.
- Optimize the code to handle more data faster. We propose to use AWS resources to parallelize even more the evaluation of our quantum circuit (not only for gradient computation)
- Improve our synthetic sentence dataset to avoid absurd examples.
- Try new types of *Shuffling* and *Entangling* layers. Allow for different type of controlled rotations (RX, RZ). Eventually add single qubit gates as well.

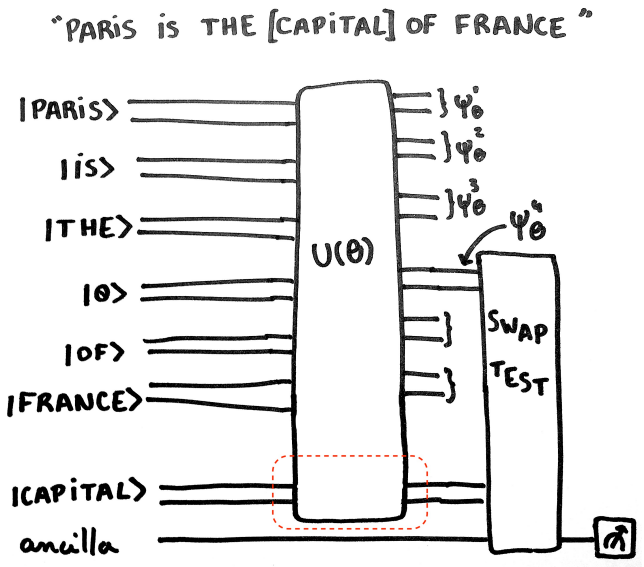

Figure 10: Porposition of a different circuit

References

[1] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi, "Foundations for near-term quantum natural language processing," *arXiv preprint arXiv:2012.03755*, 2020.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.