Engineering of a QNLP pipeline for software requirements classification

Software Engineering for Artificial Intelligence - Project Report vers. 0.2

Marco Calenda
Departement of Computer Science
University of Salerno
Fisciano, Italy
m.calenda10@studenti.unisa.it

Github Repository

***Abstract*—.**

***Index Terms*—software engineering, non-functional requirements, quantum natural language processing**

I. Introduction

In recent years, Large Language Models (LLMs) based on Self-Attention [1] – BERT or GPT-3 for instance – putted in the shade simpler neural network architecture – such as Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM) – for what concerns Natural Language Processing (NLP) tasks. Current LLMs have reached a huge amount of parameters and they require large amounts of data to function properly leading to high complexity and open researches on the interpretability of the learned information and the explainability of the model responses.

A still in its infancy alternative, gaining popularity and achievability in the last few years, consists in the exploitation of quantum for NLP tasks (QNLP) – a cutting-edge area. The main idea is to model natural language and its underlying structure combining both (i) the distributional semantics [2], relying on statistics about the contexts in which words occur, and (ii) the compositional semantics, arguing that the meaning of a sentence is the result of the meanings of the words composing it – a view more aligned with real human cognitive processes. So far, the latter, although it would steps towards a *grammar-informed* NLP, has obtained less success in real applications while the distributional approach, exploited by Machine/Deep Learning with empirical big data, is the currently pivot of state-of-the-art NLP solutions.

In the field of QNLP, the *de facto* framework used to model natural language is the compositional model of Coecke et al. [3] – often dubbed as DisCoCat[1] – which allows encoding sentences as string diagrams and monoidal categories based on the pregroup grammar formalism proposed by Lambek [4]. The underpinning mathematical structures show the computational limits of a traditional computer yet they are suitable for a translation into quantum circuits able to exponentially speed up the process [5]. Through parameterized quantum circuit learning, i.e. Quantum Machine Learning (QML), text classification tasks can be realized; however, they are still limited, by the number of required qubits, in terms of vocabulary range and sentences length [6], [7], hence, it is still challenging to scale in size as a current LLMs and prove a quantum-advantage.

In this paper, by applying a so-called *quantum-inspired* approach, we explore the QNLP potential applied into a quasi-real scenario task of software requirements engineering. During the latter development phase, one of the most time-consuming and experienced task is the classification of Non-Functional Requirements (NFRs) [8]. Unlike the Functional Requirements (FRs), centrally written and well-formed in the requirement documents, NFRs are not always explicit and, most of the times, they are scattered and concealed into natural languages sentences that are ambiguous and imprecise. Furthermore, the lack of knowledge of NFRs in the early stages of IT projects has a huge impact on the total cost and the failure rate [9], [10]. Accurately automating the process of classification of needs into FRs or NFRs may be advantageous because it would improve and speed up the whole requirements engineering process.

Therefore, this study presents Q-DELOREAN[2] a novel tool composed by (i) a classical QNLP pipeline able to shape the requirement descriptions as tensor network — following the DisCoCat framework — and (ii) a binary classifier involving a tensor-based model trained and tested on a classical hardware. Finally, we conduct an empirical study to assess the benefits and drawbacks of the proposed approach comparing it with traditional algorithms.

This paper is structured as follows: Section II introduces the goals of the study, both for the SE4AI course and the planned

[1] DIStributional COmpositional CATegorical

[2] Quantum DEep Learning fOr REquirements ANalysis

future work, and the research questions; Section III describes the QNLP pipeline and the classifier; Section IV illustrates the methodical steps required to address the research questions and meet our goals; Section V outlines the experiments, responds to the research questions and provides the findings. Finally, Section VI presents the conclusions and further works. The incomplete sections will be included in the final report scheduled for the end of May.

II. Goals and research questions

The goal of this study is to provide a practical view, considering the constraints imposed by current quantum resources, of how a compositional framework can be used to model domain-specific and technical-oriented requirements and how does a classical classifier when applied for a real scenario task of software requirement engineering – which is still not experimented in the QNLP field.

Specifically, our paper is structured around three research questions (**RQs**).

RQ$_1$. *To what extent can the proposed tool classify FRs and NFRs?*

RQ$_2$. *How does the* DisCoCat *framework compared to other compositional model?*

RQ$_3$. *How does the proposed tool compared to classical NLP models?*

The first two research questions (**RQ$_1$**, **RQ$_2$**) seeks to understand the performance of the proposed tool and its effectiveness in accurately distinguish between FRs and NFRs. Also, it is intended to provide an empirical analysis on the pros and cons of the DisCoCat framework compared to other lightweight compositional model widely used in the QNLP. Finally, it is also analyzed the impact of the parameters configuration on the performance of the QNLP solution by conducting an empirical analysis in order to get a satisfying trade-off between accuracy and computational load. The latter will concern the optimisation choices taken during the QNLP pipeline in order to make the quantum approach feasible on a classical hardware, and the model hyper-parameters tuning.

The third research question (**RQ$_3$**) allows us to compare Q-DELOREAN with classical language models, identifying the benefits and drawbacks of using a compositional approach despite a full distributional one. To address this, the computational limit emerged out from **RQ$_1$** and **RQ$_2$** plays a key role and therefore it will be taken into account when findings will be drawn. The latter purpose is to compare Q-DELOREAN with models based on full statistical words representation and word-embeddings. Therefore, it is not intended to prove that a QNLP pipeline solves the requirements classification case better than a state-of-the-art classical model, which are not in the scope of this comparison; researches have only proved the quantum-advantage theoretically but for a practical overtaking there is a need for a growth of quantum resources and their reliability – i.e. qubit stability.

Furthermore, extra work will be planned, formalized as research questions and executed in future. We aim to conduct a comprehensive analysis about the pros and cons of involving quantum circuits despite classical tensor networks, hence, exploring the QNLP capabilities. To carry this out, quantum simulations will be involved, both noisy shot-based simulation (a simulation as close as it could be on actual quantum run) and noiseless. The experiments could also explore real executions on IBM quantum computers. The latter study is also intended to execute a qualitative investigation aiming at understanding in which way a model involving both distributional and symbolic approach is more explainable than a state-of-the-art NLP model relying mostly on the statistical information of empirical data.

III. Model Development

In this section we illustrate the general pipeline of Q-DELOREAN, in particular the process of how to go from a requirement description, through the string diagram form, to its representation as a tensor network on the basis of which the model predicts if the requirements is functional or non-functional.

A. Data preprocessing

In order to prepare text data for analysis we use various pre-processing techniques of the raw text. It's crucial, since we are persuing on DisCoCat, to remove any inflections but respect the part-of-speech of the word – a not so relevant issue in a classical NLP preprocessing pipeline. The main steps are outlined below:

- Tokenization: we break down the requirment description into individual components at word level.
- Stemming: we reduce words to their root or stem form. This is done to reduce the number of unique words in order to improve the efficiency of subsequent processing. This phase, in our case, plays a key role since we are converting each unique word into a tensor.
- Lemmatization: similar to stemming yet we reduce words to their base form or lemma (dictionary form). This can result in more accurate analysis, as it takes into account the context in which the word is used.
- Stop-word removal: we remove common words, which are in the way, not carrying much meaning. For instance, in respect to the pregroup grammar, the specific word "the" returns a noun if has a noun on his right ($n\ n^l$) so it can be removed.

After this, the preprocessed sentences are translated into string diagrams and to carry this out we have employed Lambeq [11], an high-level python library which provides a transition in the closest alternative grammar formalism, namely Combinatory Categorial Grammar (CCG) [12]; due CCG parser are available, the sentence syntax tree is mapped into a string diagram which faithfully encodes the syntactic structure of the sentence. Lambeq uses DisCoPy as a backend in order to store and manipulate string diagrams, a specialised Python library designed for this purpose.

Since syntactic derivations in pregroup form can become extremely complicated, especially for requirements description which represent a real example of natural language sentences without a fixed grammatical structure, we apply to string diagrams the so-called *rewriting rules* which allow a substantial improvement for our classical experiment by reducing the order of the preposition tensor. For what concerns our specific use case, we built ad-hoc rewriting rules for typical recurrent portions of sentence in requirements description, such as "The system shall" or "The user must". Other typical rewriting rules involved in our string diagram preprocessing pipeline are:

- Removes auxiliary verbs (such as "do") by replacing them with caps[3].
- Removes sentence connectors (such as "that") by replacing them with caps.
- Simplifies "and" by replacing it with a layer of interleaving spiders[4].
- Removes determiners (such as "the") by replacing them with caps.

Finally, the string diagram is converted into a tensor network by applying an *ansatz*, a parameterisation choice which concerns the selection of the space dimensionality that every wire of the string diagram, namely the atomic groups noun (n), adjoints (n^l and n^r) and sentences (s), is associated with in the tensor network. Lambeq includes ansätze for converting tensors into various forms of Matrix Product States (MPSs), in particular, we apply a SpiderAnsatz splitting tensors with order greater than a tunable threshold into chains of lower order tensors, connected with spiders.

B. Model Training

To perform the training, Q-DELOREAN uses PyTorch as a backend, starting from a simple model performing a simple tensor contraction of the tensorised diagrams thus exploring the development of a deep learning model by adding additional custom layers. In the latter, the number of hidden layers, the neurons per layer and the activation functions are strictly problem-specific hence this part of the architecture will be thoroughly explored.

Since this is a binary classification task, we use binary cross-entropy as loss function and, as optimizer, we choose Adam with weight decay, in order to prevent the exploding gradient phenomena; this consists into adding a regularization term to the loss function that penalizes large weights in a way that they are linearly decayed toward zero. We choose the L2 term, i.e. sum the squares of the weights.

IV. Methodical steps

In this section we outline the assessment of Q-DELOREAN and the evaluation process in order to address our **RQ**s.

[3]In the context of string diagram, a cap is primarily employed to "bridge" disconnected wires in order to change the regular flow of information and compositional connection. By internally linking certain words, we simplify the tensor dimensionality and so the computations.

[4]An alternative for Frobenius algebra, a commutative operation.

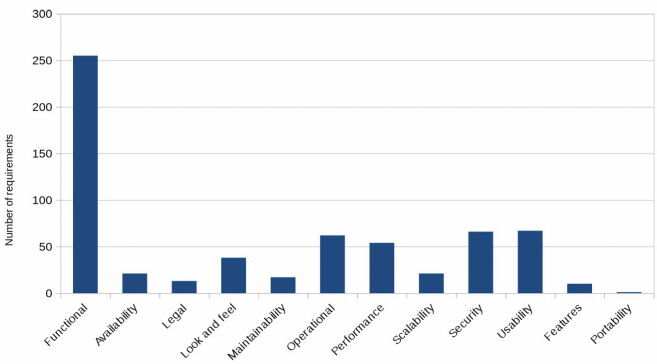

Fig. 1. Distribution of the PROMISE NFR dataset

A. Context of the study

For the assessment of Q-DELOREAN we used the PROMISE NFR dataset [13] and in particular a relabeled version provided by Dalpiaz et al. [14], a widely used dataset in the research area of software requirements classification. It consists of 625 requirement description sentences, collected from 15 different projects, among those 255 are functional requirements and 370 are non-functional requirements. The full distribution of the sub-classes is shown in Fig.1.

B. Data collection

In spite of his limited size, which would be trivial for a robust LLM in a sort of way, it represents an hard challenge for a compositional approach especially the DisCoCat framework. The latter strong dependency on pregroup grammars comes with some significant restrictions namely the absence of a pregroup parser, at least up to now, preventing large-scale experimentation and also the restriction of the expressibility of the model to context-free grammars – not suitable for natural language. As previously mentioned, this problem is somehow circumvented by the CCG translation still with sporadic cases of sentences wrongly translated in string diagram.

Nonetheless, also in order to reduce the vocabulary size and grammatical complexity, we performed a manual filtering, cutting off sentences composed by (i) words too much specific or technical, (ii) large requirement description composed by more sentences correlated and (iii) sentences with a very complex grammatical structure; the aim is to produce string diagrams with simple yet correct compositional semantics. The manual filtering is carried out always keeping the dataset well balanced.

C. Data analysis

In our experiment we follow the classic supervised learning iter, splitting the dataset into training, validation, and test sets. The validation set is used to tune the model's hyperparameters, we perform the cross-validation technique by partitioning the data into multiple subsets, or folds, and training the model on different subsets while using the remaining fold(s) for validation.

As evaluation metrics the experiments use Precision, Recall, and F1-score to measure the effectiveness of Q-DELOREAN and address in a quantitative way the research questions:

- Precision: the ratio of correctly predicted positive observations to the total predicted positive observations.
- Recall: the ratio of correctly predicted positive observations to all observations.
- F1-Score: the weighted average of Precision and Recall.

In order to answer $\mathbf{RQ}_2$, in the parsing phase of the QNLP pipeline are employed other compositional models, successively compared:

- Bag of Word (BoW): represents a sentence as a multiset of words not taking into account the order of words or any other syntactic relationship between them.
- cups reader: a basic tensor network in which all tensors have the same shape and each is connected to the next one.
- stairs reader: combines consecutive words using a box in a recurrent fashion, similarly to an RNN.
- tree reader: a CCG derivation directly interpreted as a series of compositions without a translation into a pregroup form (as DisCoCat).

To address $\mathbf{RQ}_3$ we compare Q-DELOREAN with other classical text classifiers, not state-of-the-art as before mentioned:

- TF-IDF feature extractor coupled with a Naive Bayes classifier.
- Word2vec word-embedding builder with a feed-forwarded and fully-connected Neural Network as classifier.

V. Results

VI. Conclusion and further works

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[2] Z. S. Harris, "Distributional structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954.

[3] B. Coecke, M. Sadrzadeh, and S. Clark, "Mathematical foundations for a compositional distributional model of meaning," 2010.

[4] J. Lambek, "From word to sentence: A pregroup analysis of the object pronoun who(m)," *Journal of Logic, Language, and Information*, vol. 16, no. 3, pp. 303–323, 2007.

[5] W. Zeng and B. Coecke, "Quantum algorithms for compositional natural language processing," *Electronic Proceedings in Theoretical Computer Science*, 2016.

[6] K. Meichanetzidis, A. Toumi, G. de Felice, and B. Coecke, "Grammar-aware sentence classification on quantum computers," *Quantum Machine Intelligence*, vol. 5, no. 1, 2023.

[7] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke, "Qnlp in practice: Running compositional models of meaning on a quantum computer," 2021.

[8] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are "non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 832–842, 2016.

[9] R. R. Maiti and F. J. Mitropoulos, "Capturing, eliciting, predicting and prioritizing (cepp) non-functional requirements metadata during the early stages of agile software development," pp. 1–8, 2015.

[10] V. Bajpai and R. Gorthi, "On non-functional requirements: A survey," 03 2012, pp. 1–4.

[11] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, and B. Coecke, "lambeq: An Efficient High-Level Python Library for Quantum NLP," *arXiv preprint arXiv:2110.04236*, 2021.

[12] R. Yeung and D. Kartsaklis, "A ccg-based version of the discocat framework," 2021.

[13] J. Sayyad Shirabad and T. Menzies, "The promise repository of software engineering databases." School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[14] *Requirements Classification with Interpretable Machine Learning and Dependency Parsing*, 2019.