Engineering of a QNLP pipeline for software requirements classification

Software Engineering for Artificial Intelligence - Final Report

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

Github Repository

***Abstract*—This research paper explores the potential of Quantum Natural Language Processing (QNLP) in the field of software requirements engineering. It introduces Q-DELOREAN, a novel tool that combines a classical QNLP pipeline with a tensor-based classifier for the classification of Functional Requirements (FRs) and Non-Functional Requirements (NFRs) trained and tested on classical hardware. The QNLP pipeline involves preprocessing the requirement text, parsing it into string diagrams using the DISCOCAT framework, and parameterizing the diagrams as tensor networks. The study aims to explore the classification capabilities of the proposed tool, the comparison of the DISCOCAT framework with other compositional models, and the comparison of Q-DELOREAN with classical NLP models. The empirical study conducted demonstrates the effectiveness of the tool in accurately classifying FRs and NFRs using the discocat framework showing that the grammar was an added value in this particular scenario. The comparison with classical language models reveals that Q-DELOREAN outperforms traditional approaches such as TF-IDF Naive Bayes while exhibiting similarities to Word2Vec, albeit with fewer parameters yet longer training time. Overall, this research contributes to the understanding of QNLP's applicability in real-world scenarios and lays the foundation for future advancements in the field.**

***Index Terms*—software engineering, requirements classification, quantum natural language processing, discocat framework**

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 closer view to a real human cognitive process. 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 current 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 as a current LLMs and prove a practical 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

[1] DIStributional COmpositional CATegorical

natural languages sentences that are ambiguous and imprecise. Furthermore, the lack of knowledge of NFRs in the early stages has a huge impact on the total cost and the failure rate of IT projects [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 requirements description 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 (regarding the SE4AI course and the future works) and the research questions; Section III describes the architecture of Q-DELOREAN; Section IV illustrates the methodical steps required to validate the tool, address the research questions hence meet our goals; Section V outlines the conducted experiment, responds to the research questions and provides the findings; finally, Section VI presents the conclusions and further works.

II. GOALS AND RESEARCH QUESTIONS

The goal of the 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 sentences and how does a classical tensor-based classifier when applied for a real scenario task of software requirement engineering – a still not experimented field in the QNLP.

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 models?*

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

The primary focus of the first research questions (**RQ$_1$**) is to assess the capabilities of the proposed tool and evaluate its effectiveness in accurately classifying functional requirements and non-functional requirements.

The **RQ$_2$** 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. Also, they both concern the analysis of the impact of parameters configuration by conducting an empirical analysis in order to get a satisfying trade-off between accuracy and computational load of the QNLP solution. The latter concern the optimisation choices taken during the QNLP pipeline, in order to make the quantum approach feasible on a classical hardware, and the classifier 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 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 – they 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.

III. MODEL DEVELOPMENT

In this section we illustrate Q-DELOREAN, in particular, the QNLP pipeline and 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 the requirement text for analysis we use various preprocessing techniques of the raw text. It's crucial, since we are involving also the DISCOCAT framework, to remove any word inflection but respect their part-of-speech in the sentence – a not so relevant issue in a classical NLP preprocessing pipeline. The main steps and their implications are outlined below:

- *Tokenization*: we break down the requirement description into individual components at word level throwing away certain characters – such as punctuation.
- *Lemmatization*: 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. The aim is reduce the number of unique words in order to improve the efficiency of subsequent processing, in our case, it plays a key role since we are converting each unique word into a tensor (or word-embedding for classical models).
- *Stop-words removal*: some of the most common words in English are removed as they do not carry much meaning. In the DISCOCAT framework some of the stop-words are managed in the diagram rewriting phase.

All the steps are performed using nltk and TextBlob libraries.

B. QNLP pipeline

An high-level overview of the sequence of steps involved in the QNLP pipeline is showed in Fig.1.

[2]Quantum DEep Learning fOr REquirements ANalysis

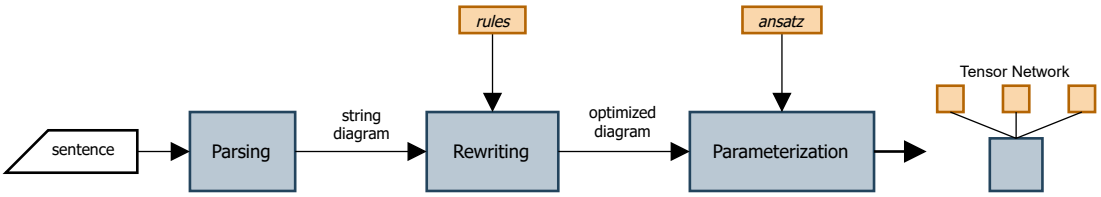

Fig. 1. Steps of the QNLP pipeline

a) Parsing: The preprocessed sentences are translated into string diagrams i.e. a diagrammatic representations of parts of text based on category theory and to carry this out we employ Lambeq [11], an high-level python library which provides a transition in the closest alternative grammar formalism, namely Combinatory Categorial Grammar (CCG) [12]; we involve the Bobcat CCG parser in order to parse the sentence and build a syntax tree, successively mapped into a string diagram which faithfully encodes the syntactic structure of the sentence. Since the text data we are dealing with is not always grammatically correct some sentences might be not correctly parsed hence discarded. In such a scenario, resorting to other linear reader proves more beneficial as they do not involve a parsing operation.

b) Rewriting: 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 the so-called *rewriting rules* able to grant us a substantial improvement for our classical experiment by reducing the order of the preposition tensor without losing its expressiveness. Most of the rewriting rules concern the internal linking of words using caps[3]. Other rewriting rules involved in the string diagram rewriting phase are:

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

It’s worth noting that rewriting phase is an optional step of the pipeline performed only for compositional model based on pregroup grammars (tree readers and DisCoCat).

c) Parameterization: Finally, we apply to the string diagram an *ansatz*, a parameterization choice that determines the dimensionality assigned to each wire in the tensor network, corresponding to the atomic groups such as nouns, sentences, conjunctions, etc., represented in the string diagram. 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 4 into chains of lower order tensors, connected with spiders. For compositional models that are not based on pregroup grammars, we employ the basic TensorAnsatz with a dimensionality of 4 for each sentence. By choosing an appropriate ansatz, we strike a balance between computational complexity and the ability to capture the compositional nature of language effectively. Parametrization is also where quantum comes into play in fact Lambeq provide ansatz to convert string diagrams into concrete parameterized quantum circuits choosing the number of qubits required for the quantum computations.

[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, hence, simplifying the tensor dimensionality.

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

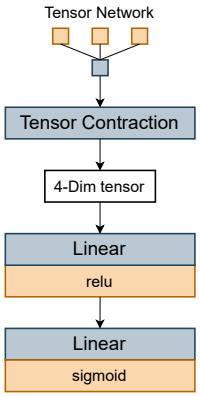

Fig. 2. Q-DELOREAN neural network

C. Model Training

We now aim at providing a model-level view, highlighting how we actually train a quantum-ready model and predict on unseen data. To perform the training, Q-DELOREAN uses PyTorch as a backend, the model (Fig.2) is composed by the following layers:

- tensor contraction layer, this operation involves combining different tensorised diagrams in a contracted 4-dimensional form using Google’s tensornetwork library.
- linear layer with relu activation.
- linear layer with sigmoid activation for binary classification.

Since this is a binary classification task, we use binary cross-entropy as loss function and in particular BCEWithLogitsLoss that combines a Sigmoid layer and the BCELoss in one single class. As optimizer we choose Adam and, in order to prevent the exploding gradient phenomena we fit out Adam with weight decay; it consists into adding a regularization term to

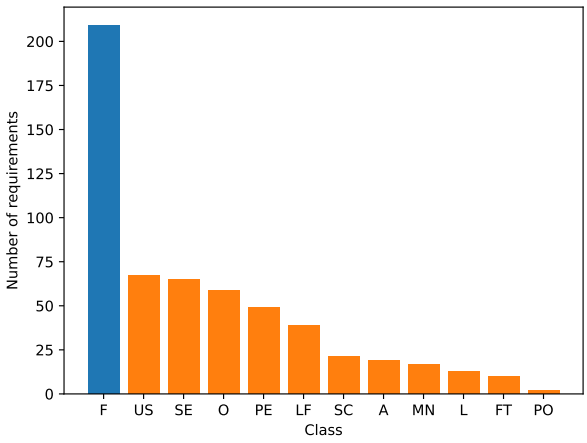

Fig. 3. Distribution of the PROMISE NFR dataset

the loss function that penalizes large weights in a way that they are linearly decayed toward zero, in our case, we choose the L2 term i.e. the sum of 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 research questions.

A. Context of the study

To conduct the evaluation, a dataset comprising a diverse range of software requirements from various domains is collected. 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 subclasses is shown in Fig.3.

B. Data collection

Despite its limited size, which may be considered trivial for a robust LLM, the dataset used in this study presents a challenging task 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 converted into string diagram or parsing failure.

Nonetheless, 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 accurate compositional semantics. It's worth noting that the manual filtering was performed while maintaining the same balance between functional and non-functional requirements in the dataset.

C. Data analysis

In our experiment we follow the classic supervised learning apporach splitting the dataset into training and test sets (80%:20%). In order to find good hyper-parameters we use a grid-search to tune batch size, optimizer's learning rate and number of neurons in hidden layers. Once the pseudo-optimal hyperparameters are identified through the grid-search process, we perform a cross-validation technique by partitioning the training data into multiple subsets, or folds, and training the model on different subsets while using the remaining fold(s) for validation. This allows us to obtain a more robust assessment of the tool's performance by reducing the impact of data variability and overfitting.

By employing cross-validation we can obtain more reliable evaluation metrics, the experiments use F1-score and accuracy 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.
- *Accuracy*: the ratio of the total number of correct predictions to the total number of observations.

In order to answer $\mathbf{RQ}_2$, during the parsing phase of the QNLP pipeline we employ other compositional models in addition to DisCoCat, successively comparing their performances:

- *Bag of Word (BoW) or Spiders reader*: represents a sentence as a multiset of words not taking into account the order of words or any other syntactic relationship between them – a naive approach.
- *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*: similarly to DisCoCat apply a CCG derivation following a biclosed form without any explicit conversion into a pregroup form.

To address $\mathbf{RQ}_3$ we compare Q-DELOREAN with other simple classical text classifiers relying on statistics namely (i) TF-IDF feature extractor coupled with a Naive Bayes classifier and (ii) Word2vec with a feed-forwarded and fully-connected neural network.

The experiments are conducted on the Google Colab Pro platform using an hardware accelerated runtime with a GPU

TABLE I
HYPERPARAMETER TUNING WITH GRID SEARCH

Learning Rate	Batch Size	Hidden Layers Size	Training Accuracy	Validation Accuracy
0.07	32	32	90.9%	81.3%
0.07	32	8	97.4%	87.0%
0.03	16	16	87.1%	79.2%
0.1	16	16	78.2%	70.8%

NVIDIA A100 Tensor Core. This platform includes all the necessary tools, libraries, and configurations required for the Q-DELOREAN tool and the evaluation process. Researchers can leverage this platform to replicate our experiments and validate our findings.

V. RESULTS

Following the DISCOCAT framework, the most meaningful results of the grid search are summarized in Table I. After choosing the best configuration we cross validate, we trained an finally we tested the model on unseen data. Q-DELOREAN demonstrated promising results in accurately classifying requirements, achieving on test set an accuracy of 87.6% and a f1-score of 87.2%, the confusion matrix is showed in Fig. 4

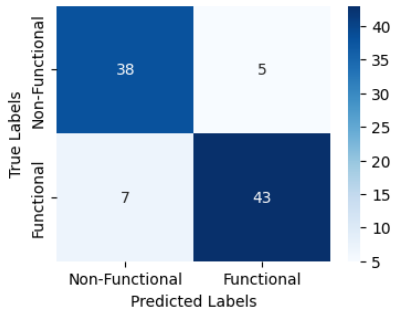

Fig. 4. Confusion matrix on Test Set using DISCOCAT

Following the evaluation of the Q-DELOREAN model using the DISCOCAT framework, we further conducted experiments utilizing other linear readers that do not involve a transformation into pregroup grammar. The results obtained from these tests revealed interesting insights into the effectiveness of various linear readers summarized in Table II. Among the various linear readers tested, it is observed that all of them exhibit lower performance compared to DISCOCAT. However, the tree reader, by performing CCG parsing, captures more semantic information of the sentence, which justifies its improved performance. In addition, by applying a comprehensive preprocessing step that further normalizes the sentence but loses some grammatical structure information, the performance differences among the various readers become less pronounced, although DISCOCAT still remains superior which shows that grammar was an added value in this particular scenario. It's worth noting that the spider reader works only with 2-dimensional sentences (due to its high computational resource requirements) hence it tends to exhibit overfitting behavior, indicating that it may not be well-suited for generalization tasks. These limitations highlight the need for further research and development to address these issues and optimize the performance of the Bag Of Words approach for QNLP.

TABLE II
READER COMPARISON

Dim	Reader	Training Accuracy	Test Accuracy
4	Stairs	80.0%	68.8%
	Cups	84.9%	76.3%
	Tree	90.2%	83.9%
	DisCoCat	97.4%	87.6%
2	Spiders	93.8%	69.2%

Finally, we compare the performances yielded by Q-DELOREAN with classical models, including TF-IDF with Naive Bayes and a Word2Vec model with a feed-forwarded neural network. The results are presented in Table III, which provides a visual representation of the accuracy on test set and number of parameters for each model. It clearly shows that Q-DELOREAN outperforms Naive Bayes and achieves similar results to Word2Vec with a significant smaller number of parameters. In the context of this experiment, apart from measuring the accuracy, we want to benchmark training and inference time of the models in order to provide further insights on this trade-off. Even though Q-DELOREAN is less complex compared to the Word2Vec model, due to its computationally intensive tensor contraction operations, the former requires more time for training compared to the latter (averaging 10 runs).

VI. CONCLUSION AND FURTHER WORKS

In this study we explored quantum-ready models for Natural Language Processing based on the DISCOCAT framework derived from the field of computational linguistics. Our study demonstrates the effectiveness of the Q-DELOREAN model obtaining overall promising results and significant insights on the usefulness of this technology for one of the most challenging field in Artificial Intelligence. By leveraging the compositional nature of language, Q-DELOREAN achieves superior performance compared to other linear readers and also it outperforms classical models like TF-IDF with Naive Bayes, while achieving similar results to Word2Vec with a neural network, but with a significantly lower number of parameters.

The findings of this study open up several avenues for future research. First, further exploration can be done to enhance the performance of the linear readers by incorporating

TABLE III
MODEL COMPARISON

Model	Number of Parameters	Test Accuracy	Training Time (15 epochs)	Inference Time (100 sentences)
Q-DELOREAN	58	87.6%	1.3 min	5 s
TF-IDF Naive Bayes	0	75.2%	-	-
Word2Vec	23k	88.2%	30 s	2 s

additional linguistic features or exploring different parsing techniques. Additionally, investigating the impact of different preprocessing strategies on the performance of the readers can provide insights into the trade-off between grammatical structure and semantic information. Furthermore, as the field of quantum computing continues to advance, exploring the potential speedup that can be achieved by implementing the DISCOCAT framework on quantum computers is an intriguing direction for future investigation. Finally, the DISCOCAT framework can be applied to other natural language processing tasks beyond requirement classification. Exploring its potential in sentiment analysis, text summarization, question-answering, or, eventually, extending this approach to other types of languages, e.g. programming languages, could prove to be effective since having have a fixed and limited grammar makes them well-suited for compositional modeling techniques like the one employed in our study.

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] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Supplementary Material for "Requirements Classification with Interpretable Machine Learning and Dependency Parsing"," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3309582