IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 June 2022, accepted 24 July 2022, date of publication 8 August 2022, date of current version 16 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3197167

RESEARCH ARTICLE

Automated Question-Answering for Interactive Decision Support in Operations & Maintenance of Wind Turbines

JOYJIT CHATTERJEE, (Member, IEEE), AND NINA DETHLEFS
Department of Computer Science and Technology, University of Hull, Hull HU6 7RX, U.K.

Corresponding author: Joyjit Chatterjee (j.chatterjee@hull.ac.uk)

This work was supported by the Department of Computer Science and Technology, Faculty of Science and Engineering, University of Hull, U.K.

ABSTRACT Intelligent question-answering (QA) systems have witnessed increased interest in recent years, particularly in their ability to facilitate information access, data interpretation or decision support. The wind energy sector is one of the most promising sources of renewable energy, yet turbines regularly suffer from failures and operational inconsistencies, leading to downtimes and significant maintenance costs. Addressing these issues requires rapid interpretation of complex and dynamic data patterns under time-critical conditions. In this article, we present a novel approach that leverages interactive, natural language-based decision support for operations & maintenance (O&M) of wind turbines. The proposed interactive QA system allows engineers to pose domain-specific questions in natural language, and provides answers (in natural language) based on the automated retrieval of information on turbine sub-components, their properties and interactions, from a bespoke domain-specific knowledge graph. As data for specific faults is often sparse, we propose the use of paraphrase generation as a way to augment the existing dataset. Our QA system leverages encoder-decoder models to generate Cypher queries to obtain domain-specific facts from the KG database in response to user-posed natural language questions. Experiments with an attention-based sequence-to-sequence (Seq2Seq) model and a transformer show that the transformer accurately predicts up to 89.75% of responses to input questions, outperforming the Seq2Seq model marginally by 0.76%, though being 9.46 times more computationally efficient. The proposed QA system can help support engineers and technicians during O&M to reduce turbine downtime and operational costs, thus improving the reliability of wind energy as a source of renewable energy.

INDEX TERMS Decision support, artificial intelligence, interactive systems, wind energy, question-answering, knowledge graphs, formal language generation, deep learning.

I. INTRODUCTION

Question-answering (QA) systems [1], [2], [3] provide intelligence analysts and other users of information systems with the ability to pose questions to a system in natural language (*semantic queries*), and obtain the relevant answers (*assistance*) they may require to better perform their tasks [4], [5], [6], [7]. QA systems have historically been successfully utilised for information retrieval in domains like e-commerce [8], education [9], tourism [10] and safety-critical applications like healthcare [11], [12], [13] etc.

QA systems leverage either pre-structured databases or a collection of domain-specific natural language documents for information retrieval [14]. In the last decade, there has been a particularly growing interest in leveraging knowledge graphs (KGs) for QA [15], wherein, various real-world entities like concepts, events, objects etc. are represented as nodes in the graph and these are interlinked via graph edges that serve as a predicate [16]. KGs can either be open-domain or domain-specific–open-domain KGs (e.g. Google KG) consist of a large collection of coarse-grained facts without being

The associate editor coordinating the review of this manuscript and approving it for publication was Longzhi Yang.

restricted to a specific domain, whereas, domain-specific KGs (e.g. in healthcare) consist of facts dedicated to specific application domains and are generally smaller in size [17]. While there has been a significant emphasis on QA in open-domain tasks, there has been limited focus on leveraging domain-specific KGs in applications aimed at AI for social good, especially pertaining to tackling climate change.

Present-day electricity systems contribute to around a quarter of human-caused greenhouse gas emissions each year [18]. To combat such emissions, there has been a growing emphasis on utilising renewable energy sources, particularly wind energy as a viable low-carbon energy resource [19], [20]. Despite their promise, wind turbines regularly suffer from operational inconsistencies that affect their reliability and availability [21], leading to significant downtimes. This ultimately affects society as turbines are not able to generate electricity at their optimal capacity, and the unreliable nature of wind energy makes many organisations reluctant to switch to low-carbon energy. Operations and maintenance (O&M) is key to prevent such inconsistencies and fix/avert failures in turbines. A key aspect of O&M is Condition-based monitoring (CBM) [20], wherein, operational changes in the turbines and their sub-components are regularly monitored for any impending faults. Traditionally, CBM has relied on signal processing and vibration analysis and more recently, AI models have been applied for data-driven decision support by leveraging the Supervisory Control & Acquisition (SCADA) data[1] generated from turbines at regular intervals [22], [23]. The wind industry has leveraged conventional machine learning (ML) models [20] as well as deep learners [24], [25] for CBM, achieving high levels of accuracy particularly in anomaly prediction and wind power forecasting.

Despite demonstrating promising applications of AI for the wind industry, existing studies fail to provide an ambient interface for automated reasoning during CBM. As *decision making time* is a critical factor for O&M activities, an automated reasoning system can provide immediate answers to engineers on *why* a fault occurs for instance, and *how* to fix/avert the failure by planning for appropriate management of SCADA parameters, alarms and condition of sub-components etc., leading to significant savings in O&M costs and reduction of downtimes and operational inconsistencies. To accomplish this objective, we propose QA over KGs, wherein, natural language questions posed by the engineers e.g. ''*What are the predictive activities for the power cabinet of wind turbines?*'' can be effectively answered by appropriate retrieval of facts pertaining to the relevant entities–which in this case would describe the ''*Predictive Activities*'' property for the entity ''*Power Cabinet*''. Note that due to the immense complexity and scale of O&M information in case of complex engineering systems like wind turbines, it is practically infeasible for engineers to remember the ''detailed'' technical facts for each and every fault type, SCADA parameters, turbine sub-component etc., which our interactive QA system can help provide through automated information retrieval from relevant domain-specific sources.

Combining language models with KGs for facilitating automated QA has historically been a highly challenging task in the NLP domain. [26]. This is particularly because: given a natural language question–*(a)* informative knowledge needs to be retrieved from domain-specific (often large) KG databases *(b)* nuances of the question and the structure of the KG need to be effectively captured to provide joint reasoning over these disparate sources of information [27]. There has been a significant research focus on training AI models for QA over KG databases across multiple application areas, particularly for open-domain tasks [17], [28], [29] by utilising natural language questions and answers that are either human-annotated or (semi)-automatically generated. Most existing studies have focused on directly generating answers in natural language itself, rather than the code which can be used to retrieve the answer from a KG. While this is an important research avenue in its own right, when applying QA to O&M of renewable energy sources in a safety-critical application (such as wind turbine maintenance), we have a need to be exact as errors in response/answer generation can have serious consequences.

While few existing studies [30], [31], [32] have demonstrated the promise of utilising KGs for systematically structuring conceptual information in the wind industry, they only serve as an end point [33] providing information for consultation without the ability to further reason over the data. Moreover, these ontologies need to be queried manually through specialised graph query languages to extract relevant and meaningful information, which may not be easily accessible to turbine engineers & technicians. To the best of our knowledge, there is no existing study which has focused on the development and utilisation of domain-specific KG databases in the wind industry wherein, KG elements are mapped to natural language questions, which can provide an easy-to-use ambient and intuitive interface for interactive decision support through automated QA. Additionally, we believe that the optimal utility of KGs for decision support in the wind industry can only be realised by integrating conceptual information on O&M with other heterogeneous data, such as SCADA parameters from sensors, alarm types, preventive/predictive/corrective maintenance action strategies for turbine sub-components etc. and their associated relationships.

We therefore opt for an approach that generates KG queries instead of direct natural language. This avoids accounting for the variability of natural language and reduces the required vocabulary (including the presence of special symbols, numbers and variables etc.). In this regard, our approach is related to work in code [34], [35], [36], [37] and formal language generation [38], [39], [40]. The retrieved answers in our study are related to look-up of information from specialised domain

[1]SCADA data from wind turbines includes continuous measurements of vital operational parameters from sensors (such as wind speed, pitch angle, active power etc.) and historical logs of alarms with natural language descriptions of their causes.

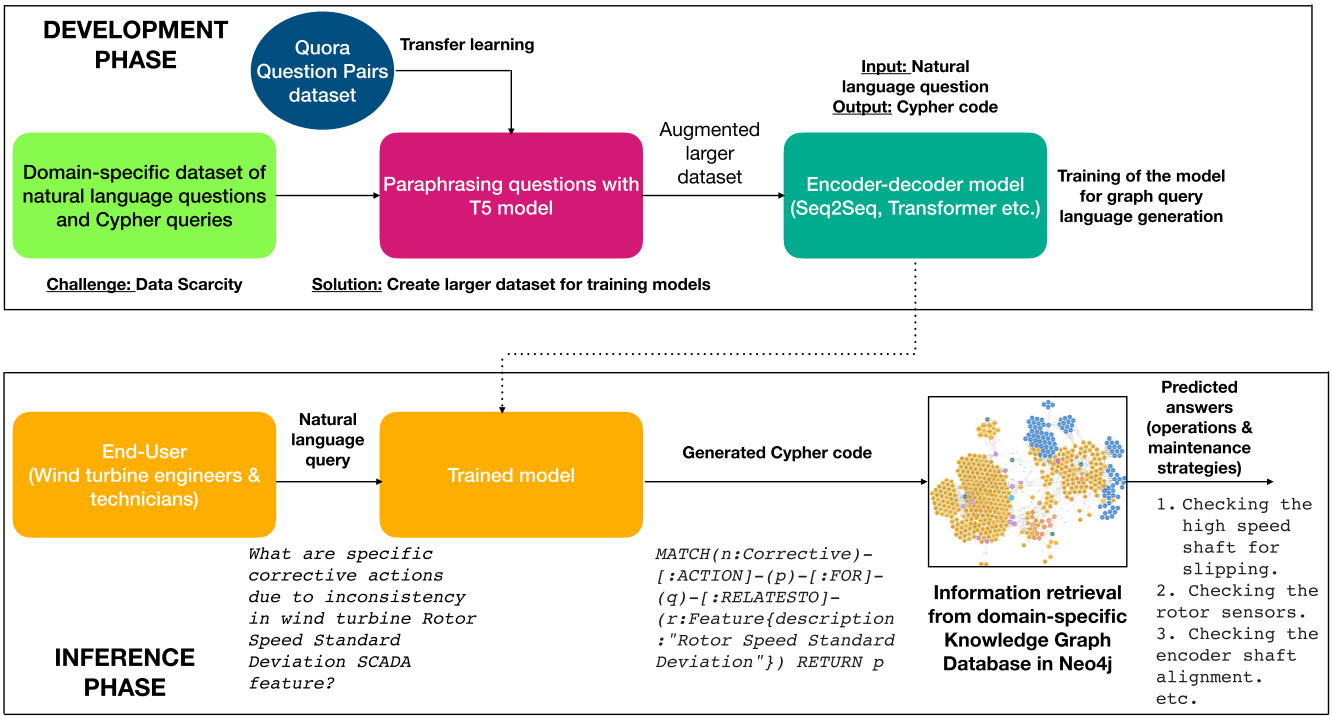


FIGURE 1. Proposed framework for graph query language generation in the wind industry, wherein, an encoder-decoder model is used to predict formal language (Cypher queries) based on domain-specific natural language english questions to facilitate automated information retrieval of O&M information. A pre-trained T5 transformer model is utilised for data augmentation through paraphrasing to tackle the sparse nature of data for specific faults in the wind industry.

resources such as user manuals for wind turbine maintenance. The proposed approach also helps the QA system to incorporate multimodal information like images of turbine subcomponents, SCADA features etc. in the responses, instead of learning to generate such entities themselves.

In this article, we propose a novel solution to multimodal decision support using a transformer model [41]. We utilise a domain-specific KG [42], [43] that we developed from human-authored maintenance manuals in the wind industry to ensure high-quality outputs. We also employ paraphrase generation for data augmentation to account for the potential variability in natural language input queries and the sparse nature of data for specific faults in the wind industry. Figure 1 depicts a conceptual overview of our proposed framework, wherein, an encoder-decoder model facilitates prediction of formal language (Cypher queries) that are utilised to automatically retrieve domain-specific O&M information based on natural language English questions. Experiments with an attention-based sequence-to-sequence (Seq2Seq) recurrent neural network model and a transformer for graph query language generation for information retrieval from the KG show that the transformer model predicts queries (and thereby the responses to natural language questions) with an accuracy of up to 89.75%.

In summary, we make the following key contributions in this paper:-

1) A novel framework for graph query language generation is proposed in a real-world application of intelligent QA systems for interactive decision support in O&M of wind turbines. The joint framework combines cross-domain NLP techniques for automatic formal language generation from varied and potentially noisy natural language inputs, and facilitates retrieval of domain-specific multimodal outputs including text, images and device measurements.
2) We develop a domain-specific dataset of natural language questions and Cypher queries for QA over a publicly available domain-specific Neo4j KG database in the wind industry. All our data is made publicly available on GitHub.[2]
3) We explore the role of pre-trained large language models for performing data augmentation of domain-specific information in the wind industry through paraphrase generation. This helps tackle the sparse nature of data for specific faults in the wind energy sector.
4) Our approach provides a complete QA system for engineers & technicians in the wind industry to automatically query domain-specific information in natural language, without requiring any specialised skills and understanding of KGs. This can potentially help reduce operational inconsistencies by assisting engineers to fix/avert failures in wind turbines in a timely manner, thereby helping make wind energy sources more reliable en-route to combat climate change.

[2]Datasets: https://github.com/joyjitchatterjee/WindTurbine-QAKG

The paper is organised as follows: Section II reviews past literature in question-answering and formal language generation. Section III describes the proposed framework for graph query language generation to facilitate automated QA for decision support in the wind industry. Section IV describes the datasets used in this study, alongside the steps used for their pre-processing. Section V provides an overview of the architecture of the Seq2Seq and transformer learning models which are utilised in this study. Section VI discusses the experiments conducted with the proposed learning models. Section VII demonstrates the experimental results obtained in generating Cypher queries with the Seq2Seq and transformer models. In Section VIII, the key takeaways from the experiments conducted in this paper and their potential utility and limitations for the wind industry are discussed. Finally, Section IX concludes the paper, enunciating the main contributions and limitations of this study and provides a path for future work.

II. RELATED WORK

In this section, we would review related work in question-answering for open and closed domains and also in the area of formal language generation. This section also discusses the limitations of existing studies and highlights the novelty of the proposed approach compared to the past literature in this domain.

A. QUESTION-ANSWER GENERATION

Domain-specific QA tasks have witnessed extensive research interest over the years, particularly in the last decade [44], [45], [46], [47], [48]. Most existing studies in this area have utilised a static lexicon to map the surface forms of the relevant entities to their logical forms [49], which makes scaling up such lexicons (that often consist of thousands of entities) challenging and inefficient. To tackle these challenges, KGs have been utilised for QA through development of *natural language interfaces to knowledge bases (NLIKB)* [39], [50], [51], [52], [53]. Existing studies performing QA over KGs have focused on utilising either general or neural network-based approaches [54].

The general QA approaches have mainly focused on leveraging information retrieval methods and semantic parsing. Information retrieval-based approaches [28], [55] analyse the dependency of words in questions to construct a candidate set consisting of possible answers retrieved from the KG, from which the most appropriate information is selected based on a quality or relevance evaluation [54]. In contrast, the studies that leverage semantic parsing [56], [57], [58], [59], [60] map natural language questions to their logical forms based on either a combinatory grammar mechanism or dependency-based compositional semantics [54]. These logical queries are then finally translated into structured queries for extracting relevant answers from the KGs [61]. Note that such structured queries (e.g. SPARQL, Cypher etc.) are essentially domain-specific code, which are executable by computers for retrieval of relevant information from the KG databases. The task of generating these queries during data-driven semantic parsing thereby pertains to *domain-specific language (DSL)*[3] generation, wherein, formalisms of the appropriate schema and syntax of code are to be learnt from the data. While the general QA approaches are simple to apply, they generally witness low evaluation scores and have mostly been outperformed by neural approaches in recent years, particularly when complex multi-hop reasoning over KGs is required.

There has been a rapidly growing interest in leveraging deep learning models for QA, particularly for open-domain tasks. This progress can be attributed to the release of large knowledge bases that consist of consolidated knowledge extracted from various sources e.g. free-form text, tables etc. (such as *Freebase*) or annotated by humans (such as *SimpleQuestions*, *WebQuestions* etc.) [28]. In a notable study in this domain [28], Memory Networks (MemNNs) have been utilised to achieve state-of-the-art performance in simple QA[4] on the WebQuestions database. MemNNs contain a memory component which can be read from or written to, along with a trainable neural network which can be used to query the memory (that can be incorporated with facts from KG databases) for appropriate information retrieval. The authors utilised the *Freebase* database as the underlying KG for the MemNN model. The paper also proposed a new *SimpleQuestions* structured KG database, and demonstrated that the model can facilitate transfer learning–thereby making high-quality predictions in a new domain (with the Reverb database) without needing to be re-trained.

There have been some other popular studies in this domain that apply various types of neural network architectures for QA: a character-level encoder-decoder model with attention [29] has been used to learn higher level semantic concepts towards answering natural language questions from KG databases. The paper demonstrated that the model, which leverages attention-based long short-term memory networks (LSTMs) for encoding and decoding, facilitates joint learning of question embeddings, entities and predicates which enables it to better generalise to unseen entities. The proposed model significantly improves the state-of-the-art performance on the *SimpleQuestions* database, while utilising significantly less data for training in comparison to previous work. Some studies [63] have utilised recurrent neural networks (RNNs) in an end-to-end manner for simple QA tasks. These studies have shown that RNNs are able to effectively model the sequential nature of the natural language questions and answers in KG databases by transforming the inputs into vectors using word or character level embeddings [17]. However, the process of modelling large amounts of sequential

[3]Domain-specific languages have code which is easy and intuitive for utilisation in a specific application domain e.g. HTML for web development, SQL for querying relational databases etc. DSLs have high-level abstractions to reduce focus on overcoming low-level challenges in programming [62].

[4]In simple QA tasks, a specific fact from the KG database would answer the user's question without requiring multi-hop reasoning over multiple facts in the KG.

data significantly increases the training time in such models, and additionally, these models are less efficient in learning long sequences in comparison to more recent neural architectures like transformers.

Attention-based convolution neural networks (CNNs) have also been explored for a similar task in answering factual questions [64], wherein, the authors show that stacking an attentive max-pooling layer over the regular convolution layers in the CNN can help to model relationships between predicates and question patterns more effectively. More recently, transformer-based models have been shown to outperform conventional end-to-end neural architectures in QA tasks [65], [66], [67], [68]. As transformers eliminate recurrence prevalent in vanilla recurrent neural networks and utilise a self-attention mechanism instead, they are generally able to learn more effectively over longer and more complex sequences of text (questions and answers) while also being computationally more efficient due to their parallelization capabilities. In Sections II-B and II-C, we would discuss past literature pertaining to a specific vein of research that leverages automatic code and formal language generation from natural language–this area provides the backbone of our proposed approach in facilitating automated QA with natural language questions through information retrieval from domain-specific KG in the wind industry.

B. AUTOMATIC CODE GENERATION

Recently, there has been a growing interest in applying deep learning to the automatic generation of high-level general purpose programming languages (such as Python, Java etc.) [34], [35], [36], [37] as a means to automatically find programs that satisfy certain criteria such as efficiency, optimality, correctness, hardware compatibility etc. This has led to a number of investigations that combine the traditionally separate fields of programming languages and natural language processing. While some studies aim to generate code directly from language (or vice versa), others make use of a formal intermediary representation, such as an abstract syntax tree (AST). Interestingly, the previous work on applying natural language processing techniques to programming code has mostly focused on the generation of small descriptions of code fragments, summaries and annotations, see e.g. the study by Haiduc *et al.* [69] for an early approach to code summarisation. Another approach to the same task [70] utilises an LSTM sequence-to-sequence model to learn questions that describe code segments from a corpus collected from StackOverflow. The authors attempt to learn a direct mapping from inputs to outputs without any intermediate representations and report low similarity with a human comparison. Hu *et al.* [71] replicate Iyer *et al.* [70]'s study and show that using an AST as input to their sequence-to-sequence learner can improve performance. Related to code summary generation, Allamanis *et al.* [72] generate method names for code snippets by learning a mapping from long input sequences to short output sequences. The authors use an LSTM and extend it with a convolutional layer that acts as a domain-invariant attention mechanism.

In terms of generating code itself, Yin and Neubig [36] trained a neural network to generate source code from natural language inputs by treating it as a semantic parsing problem. The authors demonstrate the benefit of modelling syntax explicitly and outperform previous work by 9-11%. In a related study, Ling *et al.* [73] applied latent predictor networks to generate code for a computer card game. Another study in this domain [74] utilises a hierarchical approach for neural semantic parsing across multiple tasks for generating Python source code and SQL queries. The authors generated intermediate logical forms by omitting low-level information in the code (e.g. variable names and arguments) and filling in the missing details by conditioning on the meaning representations of natural language questions. They utilised these for constructing the final logical forms using a simple encoder-decoder model.

More recent studies have focused on utilising transformers for generating code snippets from natural language descriptions. Kusupati and Ailavarapu [35] have utilised transformers for Python code snippets generation with the CoNaLa dataset, wherein, the authors utilised a modified form of back translation and cycle consistent losses to train the model in an end-to-end manner. They demonstrated that the self-attention based transformer architecture outperforms LSTM based encoder-decoder models significantly. Large pre-trained language models have also been utilised for code generation with promising results. Feng *et al.* [34] proposed a Bidirectional Encoder Representations from Transformers (BERT) based model (Code-BERT) for generating code in 6 programming languages (like Python, Java etc.). The authors also explored zero-shot learning and demonstrated that the model achieves state-of-the-art performance in generating the most semantically related code corresponding to natural language questions on the CodeSearchNet corpus, while also achieving promising performance in other downstream NLP tasks like code-documentation generation.

Overall, we observe that code generation is of growing interest to multiple communities. Current work in natural language processing focuses mostly on the analysis of code and generation of annotations rather than of code itself. Output sequences are mostly short and evaluation scores still low. The programming languages community has started to adopt neural nets but still relies mostly on engineered algorithms with a learnable component rather than fully learnt systems. In comparison to generating code in high-level programming languages, there has been limited research in generating graph query languages for querying knowledge bases.

C. FORMAL LANGUAGE GENERATION

Most early approaches to graph query language generation [38], [39] have leveraged rule-based, pattern-based or grammar-based approaches to translate natural language questions to formal queries in DSLs like SQL etc. [40]. In a notable study focusing on graph query language

generation [40], an ensemble approach has been utilised wherein, a random forest model was used for phrase mapping to identify the question types and a tree-structured LSTM model was used for SPARQL query generation. The LSTM takes into account the syntatic and semantic structure of the input questions and the tree representation of all possible candidate queries for ranking the generated queries, thereby providing the queries which are most appropriate to the questions. The method significantly outperforms the state-of-the-art QA systems on the 7th Question Answering over Linked Data Challenge (QALD-7) and the Large-Scale Complex Question Answering (LC-QuAD) databases. Some other recent studies in the area of formal language generation and KGs have explored ontology reasoning to train deep neural network models for effectively performing logical reasoning [75], generation of textual summaries from KG triples [76], learning language errors in real-word software programs [77] etc.

While there has been particularly significant research in generating SQL code from natural language questions, there is very limited research in generating Cypher queries for information retrieval in Neo4j database systems. In possibly the only study in this area, a simple yet promising approach has been presented to transform English language questions to Cypher queries for an open-domain dataset used for a university project [78]. The authors performed tokenization of the natural language string queries for assigning language tags, which were further used to perform systematic pattern matching in extracting relevant labels and variables towards generating Cypher queries. While the approach shows that a useful interface can be built to provide QA over knowledge graphs without utilising AI models by leveraging the expressive power of Cypher queries, it cannot generate more complex query patterns and understand the context of user's intentions, which are integral in real-world industrial applications. A common challenge with most existing studies is the prevalence of small datasets for QA, making it difficult to train AI models. To tackle this problem, synthetic data generation models have been explored for instance in [79], wherein, the authors utilised a fine-tuned BERT model to generate synthetic question & answer pairs. Their study demonstrated that pre-training on synthetic data helps to significantly improve the performance of the QA system on the SQuAD2 and NQ datasets. Some studies [80], [81], [82], [83] have utilised paraphrasing for data augmentation to generate synthetic data for QA. In a notable study in this domain [83], the authors experimented with different models (RNN, transformer and CNN) for generating multiple paraphrase responses for the same questions in the DBpedia KG database. The study showed that paraphrase generation can significantly improve the performance of ML models in QA over KGs, providing a more expressive QA experience. There has been rather limited application of paraphrase generation in safety-critical applications beyond existing open-domain databases, with a notable exception being [80], wherein, the authors utilised an attention-based bidirectional RNN model for clinical paraphrase generation with promising results. Additionally, the domain-specific KG databases which have previously been developed in the wind industry have also seen limited application in facilitating automatic QA.

1) LIMITATIONS OF EXISTING STUDIES

Clearly, most existing studies in generating graph query languages have focused on generating relatively simple queries for open-domain tasks, rather than for real-world safety-critical applications which generally require complex reasoning and often witness long sequences of queries with significant lexical and syntactic variations. This is particularly owed to the fact that the previous studies focusing on generating Cypher queries have not leveraged more recent advances in AI, particularly models like transformers which have shown success in other domains of automated code generation in high-level programming languages as we have discussed before.

We also observe that there is very limited research in leveraging data augmentation techniques such as paraphrasing in real-world applications, particularly for tasks wherein availability of large-scale domain-specific QA corpora is a major challenge, as in the wind industry. Thereby, we aim to leverage a small corpus of annotated data we have developed consisting of natural language questions and Cypher query pairs for fine-tuning a large pre-trained language model, and utilise the augmented data in conjunction with our domain-specific KG database to develop an automated QA system for the wind industry. We would discuss various methods for data augmentation through paraphrase generation in Section IV-C of this paper.

2) NOVELTY OF THE PROPOSED APPROACH

Note that while our proposed approach utilises existing AI models such as transformers for generating Cypher queries corresponding to natural language questions–we make a novel contribution in extending the architecture of the transformer and Seq2Seq(Att) models for performing graph query language generation, given natural language questions consisting of domain-specific data (such as SCADA features, turbine alarm records etc.) in the wind industry. This is integral for our problem–as unlike in traditional neural machine translation tasks wherein subtle inconsistencies during translation (such as while translating between languages e.g. Chinese-English) [84] do not generally affect the end retrieval of information (as despite any grammatical errors etc., the models would still generate human-intelligible information in natural language), our QC pairs are significantly complex and any major inconsistencies/errors in the generated Cypher queries can affect the ability of our proposed model to accurately retrieve relevant information from the KG (as for even minor errors in syntax/composition of the generated queries, the models would fail to retrieve any responses to natural language questions). Additionally, we also integrate the traditionally independent models with a KG database for

automated information retrieval during interactive decision support.

To the best of our knowledge, encoder-decoder models such as transformers have not been utilised for automatic QA by integrating them with multimodal domain-specific KG databases (consisting of O&M strategies towards assisting engineers & technicians) in the past in a real-world industrial context in developing expert and intelligent systems, particularly within the wind industry, as also discussed in Section II. Another novel contribution of this paper is in the data augmentation methodology we have utilised to tackle the challenge of limited data availability in the wind industry (which is also a major challenge in many other real-world application domains) by leveraging large pre-trained language models. Our research is relevant to multiple fields e.g. NLG/paraphrase generation, question-answering and interactive systems.

III. FRAMEWORK FOR INFORMATION RETRIEVAL DURING QA THROUGH GRAPH QUERY LANGUAGE GENERATION

We utilise our domain-specific KG for facilitating automated reasoning by providing engineers and technicians with a natural language interface to query the KG in Neo4j[5] (which is the world's leading open source NoSQL graph database management system widely utilised in industry) through Cypher queries (Neo4j's native graph query language) [85]. Given a natural language question, our goal is to generate the appropriate Cypher query to facilitate direct information retrieval of appropriate O&M strategies from the KG database in a fully automatic fashion. The answers in our QA system would be retrieved from the KG by mapping the Cypher queries to the corresponding details of the nodes and properties. While we specifically generate Cypher queries in this study, our approach is programming language-agnostic and can potentially be extended to other graph query languages such as SPARQL. Figure 2 shows the binned power curve [22] reflecting the operational states for an operational wind turbine rated at 7MW. As can be seen, when an anomaly occurs in the turbine's sub-components, an alarm is raised (in this example, we have labelled a pitch system alarm which causes shutdown of the turbine operation). The engineers / technicians need to take instantaneous actions to fix the alarm in a timely manner to avoid continued downtimes due to no energy production. Here, the natural language questions posed by the engineers are converted into equivalent Cypher representations with an encoder-decoder model (like transformer). These are then directly used for querying high-quality O&M actions and insights from the Neo4j Graph Database Server by leveraging the domain-specific KG that we utilise in this study.

IV. DATASET DEVELOPMENT AND PRE-PROCESSING

In this section, we describe the datasets utilised for developing the QA system. We utilise a domain-specific KG for the wind industry domain that we have previously developed from scratch based on real-world operational wind farm data and maintenance manuals [42][6] as the primary source of knowledge for information retrieval during QA. This domain-specific ontology contains 537 nodes and 1,059 relationships (of 9 distinct types), and includes various types of heterogeneous information such as descriptions of alarms, SCADA parameters, preventive/predictive/corrective O&M strategies, images of turbine sub-components etc. Note that this KG, while being a valuable source of O&M information cannot be directly utilised for QA as it does not have any labelled templates or relationships for automated reasoning based on natural language questions. To facilitate QA, we develop a specialised domain-specific corpus of natural language questions and Cypher query pairs as described below.

A. CREATION OF NATURAL LANGUAGE QUESTIONS - CYPHER QUERY PAIRS OF DOMAIN-SPECIFIC TEMPLATES

Initially, we manually created 93 pairs of domain-specific natural language questions in English and the corresponding Cypher queries required to extract the relevant answers from the Neo4j KG database. These templates are generic i.e. they do not represent the O&M actions etc. for any particular sub-component, fault type etc., but the same query can represent the relevant answers for different types of cases. As the data from wind turbines consist of several (often hundreds) of sub-components, SCADA features, alarms etc., it can be highly complex and time-consuming to manually create such natural language question-Cypher query (QC) pairs for each of the cases–thereby, we used wildcards containing specific tags (e.g. *<fevent-details>* for details of all fault types), which can later be automatically replaced with the corresponding names of the sub-components, details of the faults etc. present in the KG to develop unique QC pairs for each case. Note that in some cases, these tags were not created wherein, the QC pairs are unique and do not change across different node labels or their properties in the KG. For instance, for the question *What are the main components of the system of the wind turbine?*, there is only one unique Cypher query:-

```
MATCH(n:System)-[:CONTAINS]-(p) RETURN p
```

Table 1 describes some example tags which were created for the development of natural language template questions, alongside their descriptions. Example questions along with the relevant Cypher queries are also shown. More details on all other tags are provided in Table 2.

B. CONVERTING THE GENERIC TEMPLATES TO SPECIFIC QC PAIRS

Next, the wildcard tags in the 93 generic templates were replaced with the corresponding node labels or their attribute details, both in the natural language questions as well as the

[5]Neo4j graph database management system: https://neo4j.com

[6]XAI4Wind Knowledge Graph: http://github.com/joyjitchatterjee/XAI4Wind

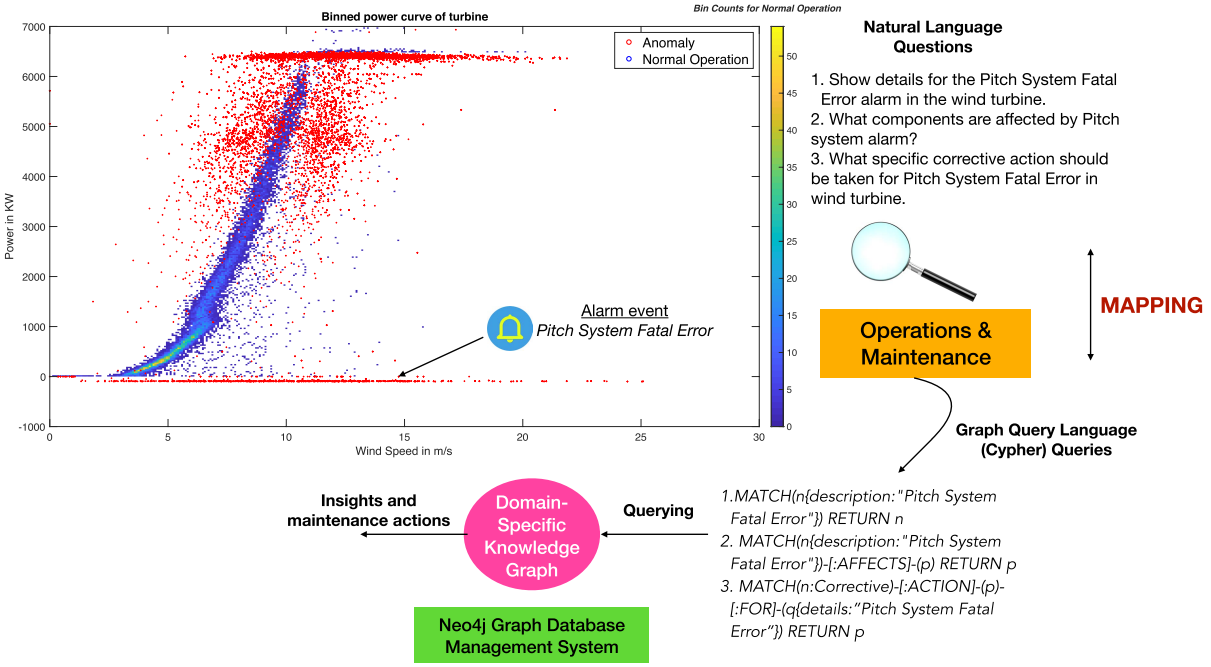


FIGURE 2. Framework for information retrieval during anomalies in turbine operation (outlined in the turbine's binned power curve). The Neo4j graph database management system is leveraged for automatically mapping natural language questions to equivalent Cypher representations generated by an encoder-decoder model (such as transformer)–facilitating automated retrieval of O&M information from the domain-specific KG.

TABLE 1. Details of some generic wildcard tags used in developing specific QC templates, along with example QC pairs–note that the tags were used to develop domain-specific natural language question templates (Q) and corresponding Cypher queries (C) to facilitate automated question-answering.

Tag	Description	Example Question (Q)	Example Cypher Query (C)
<subsys-name>	Names of 13 different types of turbine subsystems e.g. *Foundation & Concrete Section*, *Electric, Sensor & Control* etc.	Provide general predictive activities for *<subsys-name>* of wind turbine.	*MATCH(n{name:"<subsys-name>"}) RETURN n.PredictiveActivities*
<scada description>	Detailed descriptions for 102 different SCADA features e.g. *Pitch Angle Mean Value*, *Active Power Mean Value* etc.	Show specific corrective actions relating *<scadadescription>* wind turbine SCADA feature.	*MATCH(n:Corrective)-[:ACTION]-(p)-[:FOR]-(q)-[:RELATESTO]-(r:Feature{description:"<scada description >"}) RETURN p*

Cypher queries. This was done automatically based on the available details in the KG database, wherein e.g. *<subsys-name>* in the QA pairs was replaced with the names of all subsystems in the turbine, *<scadadescription>* was replaced with the detailed description of all SCADA features etc. as previously discussed in Table 1. This led us to a total of 2,361 unique QC pairs, wherein, for each natural language question, there is an associated Cypher query which extracts the relevant answers from the KG to facilitate decision support. Table 3 describes some examples of the obtained QC pairs after the replacement of the wildcard tags in the natural language question templates, along with the extracted answers from the KG.

C. DATA AUGMENTATION THROUGH PARAPHRASING

QA tasks are generally challenging due to the fact that there are various ways in which different natural language questions can express the same information need, given that there can be a large variety of surface forms pertaining to semantically equivalent expressions [81]. For instance, it is integral for a QA system in the wind industry to recognise that the natural language questions "*What are some important details for the power cabinet subsystem of the wind turbine?*" and "*What are the fundamental features of power cabinet subsystem of wind turbine?*" are completely similar in meaning despite the subtle lexical variations in these expressions. As using only a small dataset of 2,361 QC pairs described in Section IV-B for developing the QA system would significantly hamper the generalizability [86] and the system's ability to consider wider contextual information in the questions [81], it is integral to perform data augmentation towards generating a larger dataset.

A popular technique for data augmentation in developing QA systems is paraphrase[7] generation [88], [87], [86], [89], [90], wherein, texts with identical meanings are expressed in

[7]Paraphrases are defined as sentences which convey the same meaning, but have different surface realization [87].

TABLE 2. Comprehensive details of various wildcard tags used in developing QC templates, along with example QC pairs–note that the tags were used to develop domain-specific natural language question templates (Q) and corresponding Cypher queries (C) to facilitate automated question-answering.

Tag	Description	Example Question (Q)	Example Cypher Query (C)
<fng-name>	14 different Functional group names pertaining to faults in different turbine sub-components e.g. *Gearbox*, *Pitch System EFC Monitoring* etc. For normal operation of the turbine, a *No fault* functional group is used.	What are important details for the *<fng-name>* named wind turbine functional group?	*MATCH(n{name:"<fng-name>"}) RETURN n*
<scadaname>	Labelled names of 102 SCADA features e.g. *Pitch_Deg_Mean*, *Power_kW_Mean* etc.	What are details of *<scadaname>* wind turbine SCADA feature.	*MATCH(n:Feature{name:"<scadaname>"}) RETURN n*
<scada featureno>	Indices of 102 SCADA features (from *0* to *101*)	What fault events are caused by wind turbine SCADA feature number *<scadafeatureno>*?	*MATCH(n:Feature{feature_no:<scadafeatureno>})-[:RELATESTO]-(p) RETURN p*
<alarmno>	Labels for all alarms available (from *901* to *926*)	What fault events are related to alarm *<alarmno>* in the wind turbine?	*MATCH(n{alarm_no:"<alarm no>"})-[:RELATESTO]-(p) RETURN p*
<alarmdes>	Detailed descriptions of all alarms in the turbine e.g. *Wind Speed Above Max Start*, *(DEMOTED) Gearbox Filter Manifold Pressure 1 Shutdown* etc.	What components are affected by *<alarmdes>* wind turbine alarm?	*MATCH(n{description:"<alarm des>"})-[:AFFECTS]-(p) RETURN p*
<fevent-details>	Detailed descriptions of 57 different types of fault events available in the KG e.g. *Temperature measurement module failure. Error in Temperature card (RTD)* etc.	What SCADA features contribute *<fevent-details>* fault event in the wind turbine?	*MATCH(n:FaultEvents)-[:TYPE]-(p{details:"<fevent-details>"})-[:RELATESTO]-(q:Feature) RETURN q*

TABLE 3. Examples of specific natural language questions (Q) and Cypher queries (C) obtained after replacement of generic wildcard tags, along with extracted answers from the KG. These QC pairs would eventually be used to train the encoder-decoder model for automated QA.

Natural language question (Q)	Cypher query (C)	Extracted answer examples from the KG (A)
What fault events affect the wind turbine Transformer subsystem?	*MATCH(n:FaultEvents)-[:TYPE]-(p)-[:AFFECTS]-(q{name:"Transformer"}) RETURN p*	1. UPS failure. 2. Communication error. 3. Transformer fan circuit breaker. The thermal fuse of the transformer fan is deactivated. 4. Temperature measurement module failure. Error in Temperature card (RTD).
What fault events are caused in the wind turbine by the Pitch Angle Mean Value SCADA feature?	*MATCH(n:Feature{description: "Pitch Angle Mean Value"})-[:RELATESTO]-(p) RETURN p*	1. Possible existence of ice on blades. 2. Blade Position Error. 3. Pitch Activation Error. If the pitch of any of the blades differs some degrees from the reference pitch for a short period. 4. Low pitch value in stop. The position of any of the 3 blades is less than 80º in STOP mode for a specific time.
What are the specific corrective actions for High temperature on the gearbox bearing fault event in the wind turbine?	*MATCH(n:Corrective)-[:ACTION]-(p)-[:FOR]-(q{details:"High temperature on the gearbox bearing"}) RETURN p*	1. Looking for bearing damages. 2. Making sure the bearing temperature is higher than the oil one (if not wiring could be swapped). 3. Checking the multiplier pump / cooling units. 4. Checking the wiring and cables for damages.

different ways for creating variations in queries [91]. This helps to narrow down the gap prevailing between the natural language questions queried by the user and the system comprehension, increasing the likelihood of finding answers to the user's questions [91], [92].

Traditionally, paraphrase generation has mostly utilised rule based techniques [88], which perform lexical substitutions of content words [87] by leveraging resources like WordNet [93], [94]. More recently, such techniques have been significantly outperformed by neural models

utilised for paraphrase generation like Sequence-to-Sequence (Seq2Seq) [82] and transformers, which have the ability to learn long-range dependencies in the input sequences [89]. In particular, the presence of individual attention heads within transformers mimics the behaviour pertaining to the semantic and syntatic structure of sentences [41], [89], which is a key factor for paraphrase generation. There have been some other recent approaches for paraphrase generation that combine existing neural models with deep reinforcement learning techniques [90], wherein, Seq2Seq models are used as generators for producing paraphrases for given sentences, and another deep learning model is used as evaluator for judging whether two sentences are appropriate paraphrases of each other, based on which rewards are assigned for fine-tuning the generator through reinforcement learning. Other promising approaches to paraphrase generation include, for instance hybrid transformer-Seq2Seq models [89], Seq2Seq models equipped with Latent Bag of Words [87] for performing differentiable content planning and surface realization etc. However, such models generally require large amounts of training data containing original sentences alongside multiple annotated paraphrased sentences, which make them challenging to utilise when such resources are either limited or not available at all, as is the case in the wind industry.

There has been a rising interest in leveraging transfer learning techniques for tackling this problem–wherein, specialised types of large-scale pre-trained autoregressive transformer models based on the architecture of traditional neural machine translation models are used for paraphrase generation [95]. Some of the popular models which have shown success in paraphrase generation include Bidirectional and Auto-Regressive Transformer (BART), Generative Pre-trained Transformer 2 (GPT-2), XLNet, Text-To-Text Transfer Transformer (T5) etc. [96]. We chose to utilise the T5 model for our problem in generating paraphrases for our domain-specific dataset in the wind energy domain, inspired by its simplicity and the state-of-the-art performance the model has achieved in various Natural Language Processing (NLP) tasks, including paraphrase generation [97]. It should be noted that while data augmentation for tackling the challenges posed by limited data is a standard approach in many other domains, the uniqueness of our approach lies in its application to domain-specific data in the wind industry.

1) DESCRIPTION OF THE TEXT-TO-TEXT TRANSFER TRANSFORMER (T5) MODEL

The Text-To-Text Transfer Transformer (T5) [97] is a large-scale language model that adopts a unified approach in restricting the inputs and outputs to text, making it suitable for a variety of NLP tasks like document summarisation, question-answering, machine translation, sentiment classification etc., and has recently been adapted for paraphrase generation. The T5 model architecture is identical to the original transformer architecture, which consists of an encoder-decoder block with self-attention. However, an exception is that the T5 model removes the Layer Norm bias prevalent in original transformers, places layer normalisation outside the residual path and utilises a different positional embedding technique. The model is inspired by BERT's masked language modelling (MLM) objective. However, the T5 replaces multiple consecutive tokens in input sequences with a single predicted mask token, unlike BERT which utilises specific predicted mask tokens for individual words in the sequences.

The T5 model has been pre-trained on the Colossal Clean Crawled Corpus (C4) dataset, which is a large corpus (approx. 750 GB) containing clean English text that was scraped by the authors from the web. The model trained on this dataset has achieved state-of-the-art results in multiple benchmarks including text classification, summarization, QA etc. The key advantage of the model is that it treats every NLP problem as a text-to-text task, taking a text sequence as input and producing a modified text sequence as output. Note that while the original model has been shown to achieve promising results in open-domain applications, it is essential to fine-tune the model on domain-specific (or closely related) datasets depending on the downstream NLP tasks to optimally leverage transfer learning to enable the model to perform similarly (when fed with similar types of data) in these circumstances. This would eliminate the requirements of training the model from scratch, which is particularly integral in our domain of wind turbine O&M, wherein, the paraphrase generation task to be performed on our human-authored small domain-specific corpus of QC pairs significantly differs from the T5 model's original C4 corpus.

Given that the T5 model can be utilised for a variety of tasks, a prefix is incorporated in the original input sentences being fed to the model to specify the exact task which the model should perform (e.g. *translate:* $<$ *OriginalSentence* $>$ can be used to specify the model should translate from one language to another etc.).

2) UTILISING THE T5 MODEL FOR PARAPHRASE GENERATION

As our problem focuses on developing a QA system, we utilised a specialised version of the T5 model[8] which has been fine-tuned on the Quora Question Pairs dataset[9] which was originally released by Quora as a part of a 2017 Kaggle competition on recognising semantic equivalence in questions. The Quora dataset contains 404k pairs of historically marked duplicate questions, which serves the goal of obtaining paraphrases for original questions. Some recent studies have also utilised this dataset for various tasks towards natural language understanding [98], including as a part

[8]T5 model fine-tuned on the Quora Question Pairs dataset: https://github.com/ramsrigouthamg/Paraphrase-any-question-with-T5-Text-To-Text-Transfer-Transformer-

[9]2017 Kaggle Competition–Quora Question Pairs dataset: https://www.kaggle.com/c/quora-question-pairs

of the General Language Understanding Evaluation benchmark (GLUE) in the original BERT paper [99]. Note that we opted for this dataset after a careful analysis and consideration of all openly available datasets (with none presently existing in the wind industry), which showed that this was the closest match to our problem task (humans asking questions and receiving human-authored answers) towards generating diverse, human-like paraphrases.

This model has previously been successfully leveraged for data augmentation through paraphrase generation in a real-world application pertaining to the development of a human-robot interaction framework [100]. We also explored an alternative T5 model fine-tuned on the Google PAWS dataset for this task, but it was not utilised as most paraphrases generated were completely duplicate repetitions of the original questions.

Below, we describe the process utilised in generating paraphrases for our data:-

1) For the paraphrase generation task, the model takes as input a natural language question $Q = (i_1, i_2, \ldots, i_n)$, and outputs a paraphrased version of the original question as $P_Q = (o_1, o_2, \ldots, o_k) \mid \exists y_m \notin Q$. Here, there would be a maximum of k words in the generated paraphrase P_Q, which together convey a similar meaning as the source question Q. Note that the Cypher queries corresponding to the paraphrases would be exactly the same as in the original questions–it is only the natural language questions for which the paraphrases are generated in our task.
2) In line with the T5 model's requirements, a *string* prefix "*paraphrase:*" is appended at the beginning of the input questions to the model, to indicate the paraphrase generation task to be performed. Besides, an end token $< /s >$ is appended after the input question. An example input to our model is thereby of the form–"*paraphrase: What are general corrective activities for the Electric, Sensor & Control subsystem of the wind turbine?* $< /s >$". Note that each of the natural language questions in our original dataset contains the words "*wind turbine*" to ensure that the paraphrases generated by the model are specific to the wind industry.
3) We used the following model parameters–a maximum length for paraphrases generated (*max_length*) as 256 characters, a combination of top-p and top-k sampling with $top\_k = 120$ and $top\_p = 0.98$. a maximum of 50 independently sampled outputs (signifying the maximum number of paraphrases generated per input question) and early stopping to ensure that the generation terminates when the end of sentence (EOS) token $< /s >$ is reached.

With this process, we obtained a large, augmented dataset consisting of 73,105 QC pairs. As some paraphrases can be completely identical (in situations wherein the model could not generate a unique paraphrased output), we eliminated repeated versions of paraphrases, finally obtaining 72,057 QC pairs which contain only unique paraphrases.[10] The average number of paraphrases generated by the model per input question was 30.518. Figure 3 describes some key statistical metrics and linguistic features[11] (such as the various parts-of-speech, composition of unique words and symbols etc.) in the datasets before and after paraphrasing. Clearly, it can be seen that the augmented dataset obtained after paraphrasing has more variation (larger number of unique words, verbs, adjectives etc.). It is also interesting to note that interjections were not present in the original data, but are incorporated through paraphrasing. These variations in linguistic features thereby help account for the diverse nature of human language [87], which we wanted to instil into our QA system for the wind industry.

3) QUALITATIVE EVALUATION OF GENERATED PARAPHRASES

Besides the quantitative analysis above, we also performed a qualitative evaluation of the generated paraphrases by utilising Amazon Mechanical Turk (AMT).[12] In this study, humans were shown the original questions (before paraphrasing) alongside the corresponding generated paraphrases and were asked to assign ratings on a 1-5 Likert scale for semantic similarity, wherein, 1 means *completely different meaning* and 5 means they *express the same meaning*. From the larger augmented dataset obtained after paraphrasing, we randomly selected 10% of the original dataset samples before paraphrasing (236 natural language questions). For each of these 236 samples, 15 generated paraphrases were selected.

We asked 33 unique human judges to assign ratings for semantic similarity between the generated paraphrase and the original natural language question. It was also ensured that for each of the cases, two ratings are obtained from two different (unique) human judges, which led to a total of $236 * 15 * 2 = 7,080$ ratings overall. The average rating was obtained as 4.223, with a standard deviation of 1.015. Figure 4 summarises the total counts of ratings obtained across different categories on the 1-5 Likert scale. As can clearly be seen, the ratings 5 (*Exactly similar meaning*) and 4 together account for 80% of the total ratings. Also, only 2.6% of the ratings were for 1 (*completely different meaning*). These metrics clearly signify the high quality of the generated paraphrases according to human judgement.

Table 4 shows some examples of generated paraphrases obtained using the T5 model corresponding to original natural language questions before paraphrasing. As can clearly be inferred, most of the generated paraphrases are highly semantically similar to the original natural language questions (in terms of their meaning), grammatical and coherent. Additionally, clearly, there is natural variation in the generated paraphrases, reflected by the change in linguistic features and

[10] Note that in some instances, the generated paraphrases can have varying surface realizations in the form of different character cases.

[11] Obtained through the Natural Language Toolkit (NLTK) [101].

[12] https://www.mturk.com

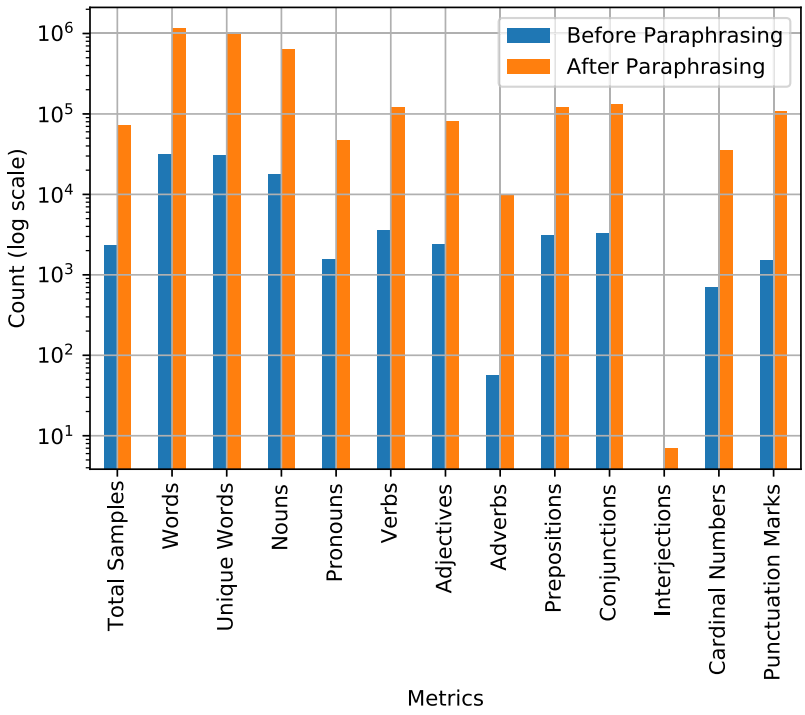


FIGURE 3. Visualisation of some key statistical metrics and linguistic features in the datasets before (original data) and after paraphrasing (augmented data)–the significant rise in the number of parameters for such metrics after paraphrasing is clearly visible, outlining the significantly increased diversity of natural language questions facilitated by paraphrase generation.

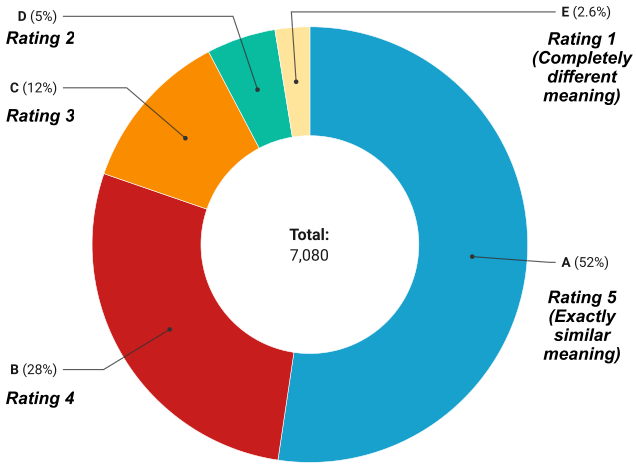


FIGURE 4. Donut chart outlining the composition of assigned human ratings in AMT on the 1-5 Likert scale for semantic similarity, based on the original questions and their generated paraphrases. Mean rating is 4.223 with standard deviation of 1.015–signifying the high quality of generated paraphrases in the augmented data based on human judgement.

surface forms (which in some cases also includes variation in the text case). However, for some instances, the generated paraphrases contain additional information which is out of context e.g. in case (b) "*in the FPGA 5M Series wind turbine VVT configuration*" is not related to the turbine under consideration for our study (7 MW Levenmouth Demonstration Turbine), but pertains to a different wind turbine model. However, it clearly does not affect the meaning in terms of the domain-specific context for the question, as for any wind turbine, the context of retrieving corrective actions pertaining to an anomaly in its *pitch angle* would be similar. For case (d), there is unexpected information pertaining to "*Brake Circuit low pressure will take 4-6 hours*", which is redundant and does not make sense based on the question's context. However, it is interesting to note that this is likely an answer to the query which is outputted within the generated paraphrase itself, and given that the original question is also present within the complete paraphrase, it does not affect the meaning and context of retrieving corrective activities for low pressure in the brake circuit of the turbine. In another instance e.g. in case (e), the word "*power*" is unexpected, as the model should have ideally generated the word "*wind*" instead to signify *wind turbine*. However, again, the context of the question remains unchanged, as it clearly enunciates the query pertaining to fault states which occur due to SCADA feature 52.

Thereby, it is clear that while there are subtle inconsistencies and errors in the generated paraphrases, as would normally be expected from a large-scale language model leveraging transfer learning on a dataset from a different domain (the Quora database), the overall nature of the generated paraphrases is promising, as our key goal is to maintain

TABLE 4. Examples of original natural language questions alongside the corresponding generated paraphrases with the T5 transformer model–note the variations in the order of words, types of linguistic features and character cases in the paraphrased questions/sentences which convey the same meaning. Any errors/inconsistencies which would not be expected based on our domain knowledge are highlighted.

Original question (before paraphrasing)	Generated paraphrases
(a) What are some important details for the Power Cabinet subsystem of the wind turbine?	1. What are some aspects of the Power Cabinet subsystem of the wind turbine? 2. What are some key details for the power cabinet of wind turbine? 3. What are the fundamental features of power cabinet subsystem of wind turbine?
(b) What are specific corrective actions due to inconsistency in Pitch Angle Maximum Value wind turbine SCADA feature?	1. What are the corrective actions due to inconsistency in Pitch Angle Maximum Value Wind Turbine SCADA feature? 2. What are the proper corrective actions due to inconsistency in pitch angle Maximum Value wind turbine SCADA? 3. What are the specific corrective actions due to inconsistency in pitch angle maximum value wind turbine SCADA feature in the FPGA 5M Series wind turbine VVT configuration?
(c) What specific preventive maintenance activities are required for the wind turbine Transformer?	1. What types of preventive maintenance activities are required for the wind turbine transformer? 2. What are a few preventive maintenance activities required for a wind turbine transformer? 3. Which preventive maintenance program is required to maintain a wind turbine transformer?
(d) Show specific corrective actions for Brake circuit low pressure failure in the wind turbine.	1.What should I do if I am still driving a wind turbine with low pressure but the Brake Circuit fails? 2. Explain specific corrective actions for Brake circuit low pressure failure in the wind turbine. 3. Show specific corrective actions for Brake circuit low pressure failure of a wind turbine. Brake Circuit low pressure will take 4-6 hours.
(e) What fault events are caused by wind turbine SCADA feature number 52?	1. How are wind turbine fault events dealt with by SCADA feature number 52? 2. What fault events are caused by wind turbine SCADA feature number 52 and how are they determined by their location? 3. What is the fault state caused by power turbine SCADA feature number 52?

the meaning of the questions within the context of the wind industry. Moreover, we observed that such major inconsistencies only occur in around 5% (3,603 cases) of the total dataset samples, which is a very minor proportion of our overall data with 72,057 samples. Additionally, some inconsistencies can actually prove worthwhile for our QA tasks, in cases wherein engineers & technicians may unknowingly input corrupted/incomplete details in the questions (e.g. incomplete alarm names, redundant information for fault events when querying corrective actions etc.). Thereby, we would utilise this augmented dataset for our further experiments in training learning models for the QA system in the forthcoming sections.

V. LEARNING MODELS

In this section, we discuss the basic architecture of the Seq2Seq and transformer learning models that are utilised in this paper. First, the concept of sequence-to-sequence code generation in encoder-decoder models is introduced. Then, we describe the principle of extending our Seq2Seq(Att) model's sequential learning process and developing the transformer model based on these descriptions. Note that while our primary learning model is a transformer, we utilise the Seq2Seq(Att) model in this paper as a baseline model for comparison of performance metrics.

A. ATTENTION-BASED SEQUENCE-TO-SEQUENCE MODEL

We propose to develop our baseline model towards generating Cypher queries as an attention-based sequence-to-sequence encoder-decoder RNN [102], [103], which would learn to condition a sequence of Cypher query fragments on the sequence of words present in a natural language question. The basic idea behind the sequence-to-sequence (Seq2Seq) model relies on leveraging a RNN for learning hidden representations $\mathbf{h}$ for an input sequence $\mathbf{x} = (x_1, \ldots, x_N)$, which it accomplishes by learning increasingly abstract encodings for the input sequences [104]. The model generates an output sequence $\mathbf{y} = (y_1, \ldots, y_M)$, which can be reconstructed based on the hidden representations $\mathbf{h}$, and $\mathbf{h}$ can be estimated based on updates computed at time step t, $\mathbf{h_t} = f(\mathbf{h_{t-1}}, \mathbf{x_t})$. Note that f here represents a nonlinear activation function (such as ReLU, tangent, sigmoid etc.). The ultimate goal in the learning process is to compress the input sequence into a single vector $\mathbf{s}$:

$$\mathbf{s} = q(\{h_1, \ldots, h_{T_x}\}), \tag{1}$$

where, q represents an activation function that is applied to the hidden representations for the input sequence $\{h_1, \ldots, h_{T_x}\}$, in order to compute the vector $\mathbf{s}$ which represents the complete input sequence transformed into a suitable range of values for the model's learning process.

During training, the aim is to minimise the loss L prevalent between the RNN's input and the desired output by computing a loss function in the Seq2Seq model's encoder-decoder architecture. Note that while multiple loss functions (such as cross entropy loss, Chebyshev loss, L1 and L2 loss, square log loss etc.) can be utilised for training deep learning models [105], [106], [107], we opted for the cross entropy loss (also referred to as negative log likelihood/loss) in line with the common practice of adopting this loss function in RNNs (which is at the core of the Seq2Seq encoder-decoder model) in the AI community [108], [103], [109], [110]. This is generally regarded as a suitable loss function, particularly in multi-class classification tasks (analogous to sequence classification in our case–wherein, for each natural language question, we aim to generate a distinct answer from a large set of various possibilities). This helps to better optimise the learning model's weights during training, unlike other loss functions like Mean Squared Error (MSE), which are generally considered to be more suitable for regression problems [111]. The cross entropy loss is computed as follows:

$$L(\mathbf{x}, \mathbf{y}) = -\frac{1}{N} \sum_{n \in N} \mathbf{x_n} \log \mathbf{y_n}, \tag{2}$$

where, N denotes the total number of training samples in the dataset, $\mathbf{y_n}$ represents the probability distribution for the predicted output sequence and $\mathbf{X_n}$ denotes the probability distribution of the original input sequence. As can be seen, log loss is used to penalise/reward the probabilities in the model's objective function based on the differences/similarities between the predicted sequence and the input sequence. These values are averaged over all N training samples in the dataset to finally compute the loss function representing the error value prevalent between the input and the predicted output sequence–$L(\mathbf{x}, \mathbf{y})$. The end goal is to maximise the probability of the model predicting individual values (representing distinct words) in the output sequence correctly.

To generate Cypher queries, we let the input sequence $\mathbf{x}$ correspond to a natural language question posed by the engineers during O&M. We assume that the output sequence $\mathbf{y}$ corresponds to a sequence of code fragments that together form a valid Cypher query, which can retrieve the most appropriate O&M responses for $\mathbf{x}$ from our domain-specific KG. An alignment vector $\alpha = (\alpha_1, \ldots, \alpha_N)$ focuses on the encoder's outputs and has the same length as the source sequence, based on which the decoder then predicts an output sequence representing a complete Cypher query that is conditioned on the context vector $\mathbf{s}$ and all previously predicted code fragments $\{y_1, \ldots, y_{t-1}\}$:

$$p(y_t|\{y_1, \ldots, y_{t-1}\}, \mathbf{s}) = g(y_{t-1}, \mathbf{h_t}, \mathbf{s}), \tag{3}$$

where g is a nonlinear function as before.

1) GATED RECURRENT UNIT AND ATTENTION

To address common problems of vanishing or exploding gradients [112], we will use a Gated Recurrent Unit (GRU) [113] for implementation of our model. A GRU computes $\mathbf{h}$ under consideration of two gates which play an integral role in controlling the model's loss and incorporation of information: the "update gate" $\mathbf{z_t}$ and "reset gate" $\mathbf{r_t}$, leading to an updated computation of $\mathbf{h}$ at time t as:

$$\mathbf{z_t} = \sigma\left(\mathbf{W_z} \cdot \left[\mathbf{h_{t-1}}, \mathbf{x_t}\right]\right) \tag{4}$$

$$\mathbf{r_t} = \sigma\left(\mathbf{W_r} \cdot \left[\mathbf{h_{t-1}}, \mathbf{x_t}\right]\right) \tag{5}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W} \cdot \left[\mathbf{r_t} * \mathbf{h_{t-1}}, \mathbf{x_t}\right]\right) \tag{6}$$

$$\mathbf{h_t} = (1 - \mathbf{z_t}) * \mathbf{h_{t-1}} + \mathbf{z_t} * \mathbf{h_t} \tag{7}$$

Here, σ denotes the logistic sigmoid function.

Finally, we integrate an attention mechanism [114], [115] which allows the decoder to access the input sequence during decoding, leading to an updated decoder:

$$p(y_t|\{y_1, \ldots, y_{t-1}\}, \mathbf{x}) = g(y_{t-1}, \mathbf{h_t}, \mathbf{s_t}), \tag{8}$$

where $\mathbf{h_t}$ is the hidden state at each time step t. In this way, each output decision is conditioned on a distinct context vector $\mathbf{s_t}$, which is specific to the decision, not the entire input sequence. Additionally, the attention mechanism provides added transparency by identifying the specific words in the input and output sequences which are important for the model's prediction. Figure 5 shows the basic structure of the Seq2Seq(Att) model utilised in our study, wherein, the GRU-based encoder-decoder architecture converts natural language questions into corresponding formal language (Cypher query) representations that can be utilised for information retrieval from the domain-specific KG database.

B. TRANSFORMER MODEL

More recently, Transformers [41] have shown prominence as a dominant architecture utilised for a variety of tasks in the AI community, outperforming conventional Seq2Seq models in multiple areas like neural machine translation, natural language understanding, question-answering etc. [116]. During its learning process, the transformer's key essence is in eliminating recurrence and convolutions, and instead computing attention weights over input sequences by utilisation of positional embeddings. The conventional attention mechanism which recurrent models utilise (wherein, the model's output sequences attend to the corresponding inputs) is extended via self-attention, which enables the inputs and outputs to both attend to themselves as well as the target sequences attending to the source [117]. Due to elimination of recurrence, the computational cost in training the model is also lowered significantly, which makes transformers highly promising for use in real-world and real-time domain-specific applications, including for QA.

Similar to the Seq2Seq model described above, the transformer also includes an encoder and a decoder. However, there are multiple attention heads prevalent in the model (referred to as multi-head attention), which constitute the essence of the model's performance and learning process:-

- Encoder's self attention, wherein the source sequence attends to itself.

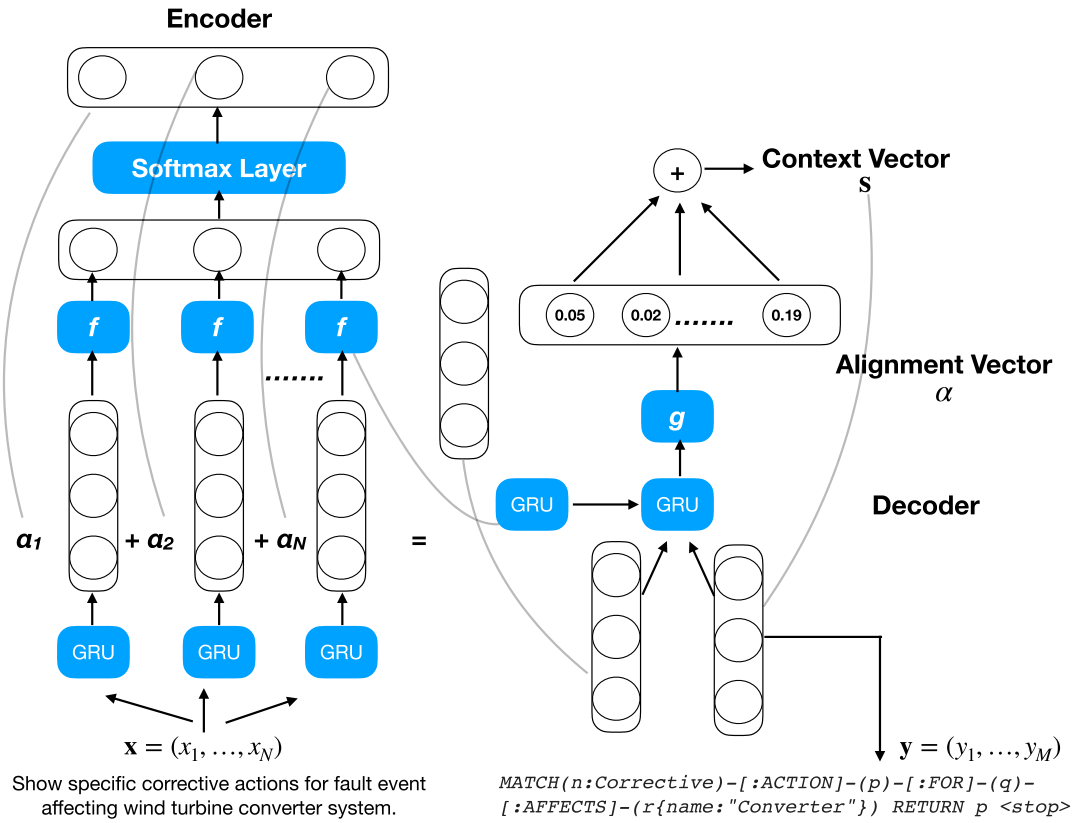


FIGURE 5. Basic structure of the Seq2Seq(Att) model utilised for graph query language generation–given a input sequence of words in a natural language question, the GRU-based encoder-decoder architecture would generate the appropriate Cypher queries for information retrieval from our domain-specific KG database.

- Decoder's self attention, wherein the target sequence attends to itself.
- The target attending to the source sequence (i.e. conventional attention as in the Seq2Seq model).

Below, we discuss the encoding and decoding stages of the transformer model briefly.

1) ENCODING

The encoder module consists of three stages, wherein, the first stage projects the input sequence into vector space. Positional embeddings of the input sequence are incorporated to record token positions in the input and account for the absence of a recurrent neural network. Positional embeddings PE are computed as:

$$PE_{(pos,2\,i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \tag{9}$$

and

$$PE_{(pos,2\,i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right), \tag{10}$$

where d_{model} denotes the depth of the transformer.

The multi-head attention stage is directed by three key parameters–*Query* Q, denoting the component that pays attention and represented as a vector of the semantic tokens corresponding to a specific input word, along with key-value pairs consisting of all the words in the sequence, represented as *Key* K and *Value* V–which signify all the initial input word vectors which the model ends up attending to. For encoding, V corresponds to the same word sequence as Q. The overall goal is to compute a weighted sum over values, where the weights assigned to individual values are computed through a compatibility function of the query with the corresponding key:

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{11}$$

where $\sqrt{d_k}$ is a scaling constant equal to the square root of the dimension of keys. QK^T computes a similarity matrix between queries Q and keys K, which is applied to the source sequences during encoding, but Q is drawn from the target sequences at the decoding stage. A position-wise feedforward neural network then identifies the input projections which should be eventually used at the decoder end.

2) DECODING

The decoder module contains five discrete stages, of which the initial two are the same as in the encoder (with embeddings offset by one position). However, masked multi-head self-attention is used instead which restricts the decoder to only focus on past words in the sequence and prevents it from looking ahead. At the third stage, the multi-head attention

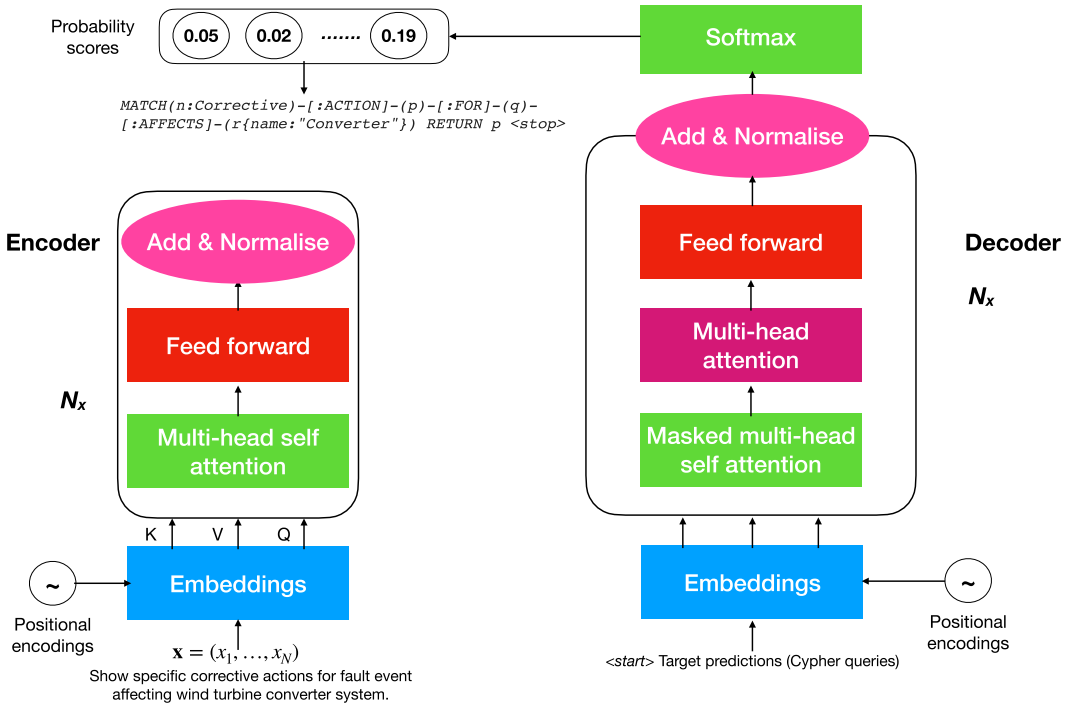


FIGURE 6. Architecture of the transformer model (adapted from [41]) utilised for graph query language generation in our QA system–the self-attention mechanism facilitates learning of Cypher query representations from natural language questions in the absence of recurrence prevalent in the conventional Seq2Seq architecture.

block captures past and final hidden representations that are received from the encoder. The fourth layer is similar to the encoder's feedforward network, and finally the softmax layer is utilised for computing the scores of words to be ultimately predicted in the target sequence.

In this paper, our goal is to extend the original transformer architecture towards the domain-specific application of QA in the wind industry. We aim to realise this by utilising the sequence of words in a natural language question $\mathbf{x} = (x_1, \ldots, x_N)$ as input to the model, and generating the appropriate Cypher query $\mathbf{y} = (y_1, \ldots, y_M)$ which is most appropriate to the context of domain-specific information requested by the turbine engineers & technicians. Additionally, similar to the Seq2Seq(Att) model, we aim to appropriately leverage the attention weights of the transformer model to gain insights into the model's decisions besides making accurate predictions of Cypher queries. Figure 6 provides an overview of the transformer architecture that we utilise in our QA system, wherein, the self-attention mechanism is utilised to facilitate sequence-aligned learning in absence of recurrence prevalent in the conventional Seq2Seq model.

VI. EXPERIMENTS

For our experiments, we utilise the data of 72,057 QC pairs obtained after paraphrasing, as described in Section IV for training the learning models. Initially, we performed word-level tokenization of the QC pairs, filtered on whitespace characters. We did not use lower-casing during tokenization as Cypher queries are case-sensitive. All numbers, special symbols and punctuations were retained in the tokenized data to ensure the generation of valid Cypher queries. Additionally, an *[UNK]* token was incorporated to account for words that fall out of vocabulary. A $< start >$ and $< stop >$ token were also added to the original samples for helping the models to clearly understand when to start and stop predicting.

We utilised a train-test split ratio of 70-30%, ensuring that the dataset is split into 50,439 training and 21,618 test instances. A batch size of 128, 256-dimensional word embeddings, input vocabulary size of 26,034 words and target vocabulary size of 1,181 words were used for training all models for 50 epochs. All models were trained in TensorFlow (Python) [118] with Adam optimisation. Table 5 describes the key experimental parameters and values which were used for training the Seq2Seq and transformer learning models in this study. The interested reader is referred to the TensorFlow documentation [13] for more details on these parameters.

- **Seq2Seq(Att):** The Seq2Seq model with GRU and Bahdanu attention, see Section V-A for details. We experimented with 512/1,024 hidden units, 2 hidden layers and a learning rate of 0.01.
- **Transformer:** The transformer model with multi-head attention mechanism, refer to Section V-B for details. We experimented with model sizes of 128/256, 2/4 total layers (multi-head attention + feed-forward layers) and

[13]TensorFlow documentation: https://www.tensorflow.org/guide

TABLE 5. Details of key experimental parameters and values utilised for training the Seq2Seq and transformer learning models.

Experimental Parameter	Value
Random State	123
Test Size	0.30
Batch Size	128
Word embedding size	256
Total epochs	50
Input vocabulary size	26,034
Target vocabulary size	1,181
Seq2Seq model Buffer Size	50,439
Input tensor (train) dimensions	(50439, 115)
Input tensor (test) dimensions	(21618, 115)
Target tensor (train) dimensions	(50439, 28)
Target tensor (test) dimensions	(21618, 28)
Dropout rate	0.10
Seq2Seq model's GRU – recurrent kernel weights initialiser	Glorot uniform
Seq2Seq model's Adam optimiser – Learning rate	0.001
Seq2Seq model's Adam optimiser – Initial decay rate for first moment of gradient (β_1)	0.9
Seq2Seq model's Adam optimiser – Initial decay rate for second moment of gradient (β_2)	0.999
Seq2Seq model's Adam optimiser – ϵ threshold	10^{-7}
Transformer model's Adam optimiser – Learning rate scheduler warmup steps	5,000
Transformer model's Adam optimiser – Initial decay rate for first moment of gradient (β_1)	0.9
Transformer model's Adam optimiser – Initial decay rate for second moment of gradient (β_2)	0.98
Transformer model's Adam optimiser – ϵ threshold	10^{-9}

2/4/8 attention heads. The model's learning rate was decayed with the *WarmupThenDecaySchedule* class in TensorFlow with 5,000 warmup steps.

To retrieve the relevant responses/answers from our domain-specific KG, we utilised *Py2neo*,[14] a specialised Python library which facilitates interfacing of the Neo4j KG database server with Python applications. Given a natural language question posed by the user, the predicted Cypher queries are automatically executed in Neo4j and O&M actions retrieved, which ultimately provides an environment for automated reasoning in our QA system.

VII. RESULTS

In this section, we discuss the experimental results obtained in generating Cypher queries with the Seq2Seq(Att) and transformer models. We also provide a qualitative evaluation of the generated queries and perform an error and output analysis for each model. Some example cases of retrieved responses from the KG based on the Cypher queries predicted by the models corresponding to natural language questions are also discussed.

A. OBJECTIVE EVALUATION

Tables 6 and 7 show the performance metrics for objective evaluation of the Seq2Seq(Att) and transformer models respectively in terms of the percentage of Cypher queries that are correctly predicted and computation time. We can clearly see that the transformer outperforms the Seq2Seq(Att) model in terms of percentage of Cypher queries correctly predicted, achieving an accuracy of up to 89.75%. The Seq2Seq(Att) model attains the highest accuracy of 88.99%, which is 0.76% worse than the transformer model. This is, although a very minor improvement in the model's performance.

Another important metric we would like to reflect on is the computation time. We note that the transformer model is the fastest, achieving the shortest computation time (20 min 34 sec). On this front, the best-performing Seq2Seq(Att) model takes 3 hr 14 min 38 sec (946.35% or 9.46 times more than the best-performing transformer). Note that these computation times were obtained with the NVIDIA Tesla K80 GPU based on Google's Compute Engine. We also observe that scaling up the transformer model (including the model size and number of attention heads) degrades the model's performance while leading to increased computation time. It is likely that more than two attention heads are not integral for learning good representations of the Cypher queries in our dataset. Given the mostly common syntax and structure of Cypher queries (e.g. all queries use common words like *MATCH, WHERE, RETURN* etc.), besides learning to generalise to intricacies of the graph query language, the models only need to learn the unique domain-specific information (e.g. alarm types, SCADA feature names etc.), rather than a large-scale vocabulary and long sequences generally prevalent in machine translation tasks. This is likely the reason why the transformer only has an improvement of 0.76% over the Seq2Seq(Att), clearly indicating that the Bahdanu attention in our Seq2Seq(Att) model suffices for the learning task if we ignore the marginal gain in accuracy.

1) CONSIDERING THE ENVIRONMENTAL IMPACT OF OUR LEARNING MODELS

As we have discussed above, the computation time for the best-performing Seq2Seq(Att) model (with 1,024 hidden units) is more than nine times that of the transformer and

[14]Py2neo Information Handbook: https://py2neo.org/2020.1/

TABLE 6. Performance metrics for Cypher queries predicted by the **Seq2Seq(Att)** model–average computation time per epoch is shown in brackets. The best performing model is outlined in bold face. Note that the highest accuracy obtained by the Seq2Seq(Att) model is 88.99% based on number of Cypher queries in the test data which are correctly predicted.

Hidden units	Total samples correctly predicted	Accuracy	Computation time
512	19,106/21,618	88.38%	**1 hr 29 min 14 sec (1.784 min)**
1,024	19,238/21,618	**88.99%**	3 hr 14 min 38 sec (3.892 min)

TABLE 7. Performance metrics for Cypher queries predicted by the **transformer** model–average computation time per epoch is shown in brackets. The best performing model is outlined in bold face. Note that the highest accuracy obtained by the transformer model is 89.75% based on number of Cypher queries in the test data which are correctly predicted.

Model size	Attention heads	No. of layers	Total samples correctly predicted	Accuracy	Computation time
128	4	4	17,655/21,618	81.66%	38 min 32 sec (46.24 sec)
	2		18,461/21,618	85.396%	33 min 7 sec (39.74 sec)
		2	**19,403/21,618**	**89.75%**	**20 min 34 sec (24.68 sec)**
256	8	4	17,859/21,618	73.36%	1 hr 29 min 7 sec (1.782 min)

for the smaller Seq2Seq(Att) model (with 512 hidden units), the computation time is more than 4.5 times as the transformer. More notably, all our transformer model configurations achieve lower computation time than the Seq2Seq(Att) model, which is in line with the parallellization capabilities of transformers on GPUs. While our models are not exponentially large given the limited availablility of data at present in the wind industry, we believe training AI models with larger datasets in the wind energy sector could make the computation time scale exponentially, particularly as new datasets focusing on O&M continue to become available, newer turbines are deployed and older turbines in operation record more data every day from their sensors.

Some recent studies [119], [120], [121] have highlighted the environmental impact of deep learning systems, wherein, scaling up the model size can have serious negative environmental consequences through its resource consumption. Moreover, these studies highlight the importance of prioritising energy efficiency for reducing negative environmental impact and inequitable access to resources [119]. As our key goal in this paper is to leverage AI in helping make wind energy sources more reliable towards tackling climate change, we echo the importance of considering dangers of rising carbon emissions over marginal improvements in performance of large language models. Converse to our results, even if the Seq2Seq(Att) model had achieved a marginal improvement over the transformer, we believe it would be the most rational decision to discard the marginally better Seq2Seq(Att) model in favour of the transformer's computational efficiency. As the transformer model meets both these expectations (best performance as well as lowest computation time), we believe they are highly promising for utilisation in the wind industry for real-time decision support when considering the *AI and Society* perspective.

B. ERROR AND OUTPUT ANALYSIS

Table 8 shows some examples of Cypher queries predicted by each of the models alongside the expected queries corresponding to natural language questions–any errors in predictions (deviation from expected Cypher queries) are highlighted, and missing words, symbols or numbers are denoted with $ symbol. Note that our QA system is case-insensitive, and any capitalisation of component names (e.g. Yaw), SCADA feature labels (e.g. ReactivePower_kVAr_Max) etc. in the natural language question examples shown only reflect the test data sample variations (including in semantic structure, linguistic features and word-cases) which were introduced during paraphrasing–the model would work the same way and generate the same Cypher queries if different case words are passed during inference.

We discuss some notable cases below for which a qualitative analysis of the predictions. Few examples outlining the visualisation of retrieved O&M information (nodes and properties) from the KG are also discussed:-

- In case *(i)* of Table 8, the natural language question pertains to retrieving corrective O&M activities for a fault event in the yaw motor. The Seq2Seq(Att) and transformer models both predict the Cypher queries correctly in this case. Figure 7 shows the output of executing the predicted Cypher query in this case pertaining to the yaw system's motor fault. As can be seen, given that the Cypher query is correctly predicted corresponding to the domain-specific natural language question, the information retrieved from the KG is also accurate. Additionally, a visual depiction of the relevant nodes and properties in the KG relevant to the question is also obtained, facilitating intuitive decision making for turbine engineers & technicians.

TABLE 8. Examples of Cypher queries generated by the Seq2Seq(Att) and transformer models –errors in predictions (deviation from expected Cypher queries) are highlighted, and $ denotes missing words, symbols or numbers in the predicted Cypher query.

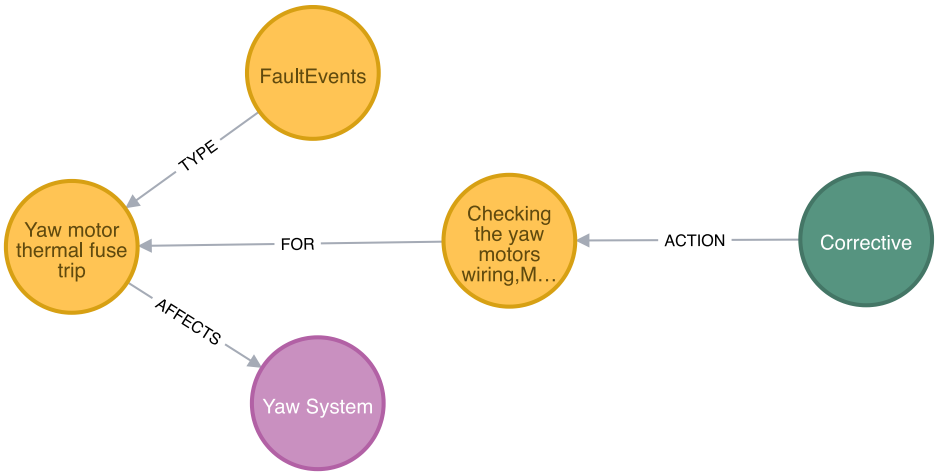
Natural language question	Expected Cypher query	Predicted Cypher query with Seq2Seq(Att)	Predicted Cypher query with Transformer
(i) What are the corrective actions for Yaw motor thermal fuse trip fault event?	*MATCH(n:Corrective)-[:ACTION]-(p)-[:FOR]-(q{details:"Yaw motor thermal fuse trip"}) RETURN p*	As expected	As expected
(ii) Which SCADA features contribute to high temperature on the gearbox oil fault event in the wind turbine?	*MATCH(n:FaultEvents)-[:TYPE]-(p{details:"High temperature on the gearbox oil"})-[:RELATESTO]-(q:Feature) RETURN q*	As expected	As expected
(iii) What is the effect of an alarm on the wind turbine yaw system?	*MATCH(n)-[:AFFECTS]-(p{name:"Yaw System"}) RETURN n*	*MATCH(n${name:"Yaw System"}) RETURN n*	As expected
(iv) What are details of ReactivePower_kVAr_Max wind turbine SCADA feature?	*MATCH(n:Feature{ name:"ReactivePower _kVAr_Max"}) RETURN n*	As expected	As expected
(v) What are fault events related to alarm 923 in the wind turbine?	*MATCH(n{alarm_no: "923"})-[:RELATESTO]-(p) RETURN p*	*MATCH(n{alarm_no: "922"})-[:RELATESTO]-(p) RETURN p*	As expected
(vi) What causes the fault events of the Absolute Wind Direction maximum value SCADA feature?	*MATCH(n:Feature{ description:"Absolute Wind Direction Maximum Value"})-[:RELATESTO]-(p) RETURN p*	As expected	As expected
(vii) What are some examples of SCADA causes of blade positioning error in wind turbine?	*MATCH(n:FaultEvents)-[:TYPE]-(p{details:"Blade Position Error"})-[:RELATESTO]-(q:Feature) RETURN q*	*MATCH(n:FaultEvents)-[:TYPE]-(p{details:"Generator-rotor speed discrepancy"})-[:RELATESTO]-(q:Feature) RETURN q*	As expected
(viii) What are the predictive activities for the Power Cabinet of Wind Turbines?	*MATCH(n{name:"Power Cabinet"}) RETURN n.PredictiveActivities*	*MATCH(n{ name:"Generator"}) RETURN n.PredictiveActivities*	*MATCH(n:Predictive)-[:ACTION]-(p)-[:FOR]-(q{name:"Power Cabinet"}) RETURN p*

- In case *(iii)* of Table 8, the natural language question focuses on determining the effect of an alarm event on the turbine's yaw system. While the transformer model correctly predicts the Cypher query in this situation, the Seq2Seq(Att) model only predicts the query partially, missing out on the *AFFECTS* relationship. This a critical error as the incorrect Cypher query would provide details of the yaw system of the turbine itself, rather than the operational inconsistencies/errors which take precedence due to an alarm in the yaw system. Figure 8 outlines the retrieved O&M actions from the KG in this case pertaining to the yaw system alarm with the transformer model, which can help engineers and technicians to understand (and fix/avert) the specific fault events and anomalies which contribute to the alarm.

  To inspect and analyse the working of each of the models during the prediction making process and the likely cause(s) of any errors, we visualise the attention weights of the models. Figures 9 and 10 [15] show the attention weights for the Cypher queries predicted by the Seq2Seq(Att) and transformer models respectively. Note that the heatmap should be interpreted based on the colors shown–darker colors denote higher attention weights (showing that the model puts greater focus on the corresponding keyword for predicting the Cypher query), while lighter colors represent lower attention weights (showing that the model places lesser emphasis on the corresponding keyword for predicting the Cypher query). Thereby, a darker color corresponding to a specific keyword in the heatmap would show that the corresponding word in the input sequence is considered more relevant/important for prediction making, while a lighter color would show that the corresponding word in the input sequence is less relevant/important to the predicted fragments in the generated Cypher queries.

  As can clearly be seen, the highest weights (activations) of the Seq2Seq(Att) model are attributed to the word *system* for predicting the *MATCH* fragment of the node. However, the keyword of *alarm* (which is the essence of the question) is focused on the node's label *n* rather than the *Yaw* itself wherein the alarm occurs. On the other hand, the transformer has a significant attention weight on the word *alarm* when it is predicting the *MATCH* segment of the relationship correctly that includes the *AFFECTS* relationship, which is essential as the query corresponds to retrieving the inconsistencies/errors which arise out of the yaw system

[15] Darker colors signify higher attention weights (importance/focus) placed by the model when making the prediction.



Examples of retrieved answers

1. Checking the yaw motors wiring.
2. Megger Yaw Motors.
3. Checking the yaw sensor.
4. Reseting the thermal breaker.
5. Ensuring the breaker amps set to the proper amps.

FIGURE 7. Output on executing the predicted Cypher query by the Seq2Seq(Att)/transformer models in case *(i)* of Table 8–examples of retrieved O&M actions for yaw motor fault event based on identified nodes and properties in our KG database are also shown. Engineers and technicians can leverage the intuitive visualisation and summary of O&M actions in this case to fix the yaw motor fault.

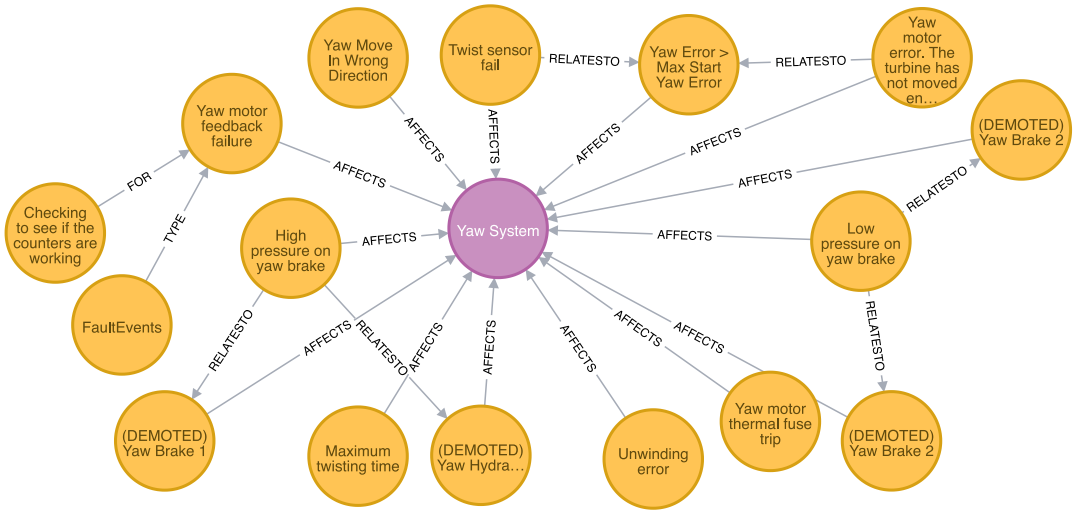


FIGURE 8. Output on executing the Cypher query predicted by the transformer model in case *(iii)* of Table 8–the nodes shown denote the inconsistencies/errors caused due to yaw system alarm. Engineers and technicians can leverage the intuitive visualisation of nodes and properties retrieved from the KG in this case to analyse (and tackle) the specific fault events and anomalies in the turbine which lead to the yaw system alarm.

alarm. Besides, note that when the end of the question is reached, the transformer places a significant emphasis on the node's label *n* whereas, the Seq2Seq(Att) model only focuses on the initial *MATCH* code fragment. This visualisation further suggests that the Seq2Seq(Att) model is lacking in its ability to focus on the keywords in natural language questions (e.g. *alarm* in this case) when predicting the code fragments, whereas, the transformer

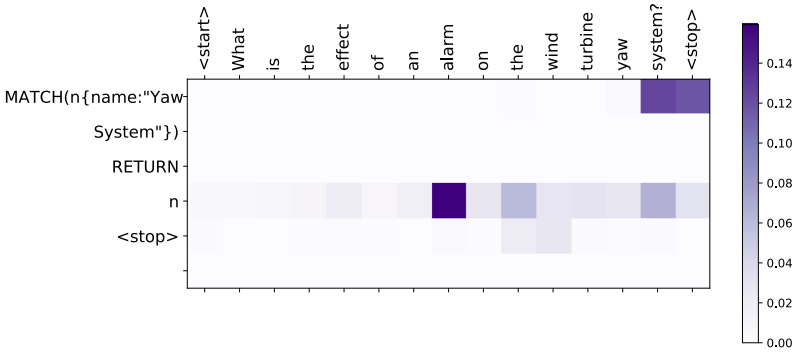


FIGURE 9. Heatmap visualisation of attention weights for prediction of Cypher query in case *(iii)* of Table 8 by the Seq2Seq(Att) model–note the *AFFECTS* relationship which the model misses by failing to focus on the question's essential keyword *alarm* when predicting the initial *MATCH* fragment.

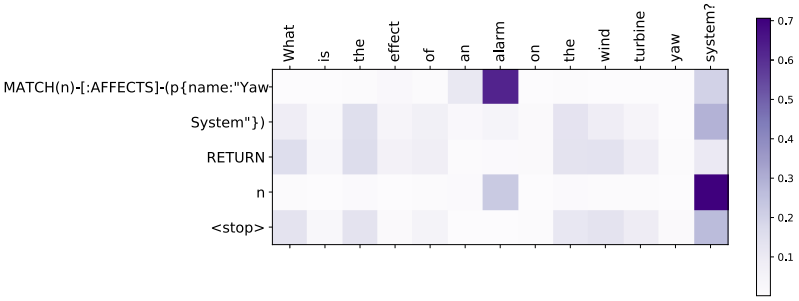


FIGURE 10. Heatmap visualisation of attention weights for prediction of Cypher query in case *(iii)* of Table 8 by the transformer model–note that the model focuses on the keyword of the question *alarm* when predicting the *MATCH* fragment and *system* when predicting the appropriate node label *n*, thereby generating the complete Cypher query correctly.

performs suitably in identifying the keywords, thus leading to the correctly generated Cypher query.

- In case *(vii)* outlined in Table 8, the Seq2Seq(Att) model fails to predict the complete Cypher query pertaining to identifying SCADA features which cause a blade positioning error in the turbine. The Seq2Seq(Att) model incorrectly references the blade positioning error due to a discrepancy in the generator rotor speed, which has no relation whatsoever with the turbine's blade position. The transformer is able to predict the complete valid Cypher query in this case. Figure 13 depicts the output of execution of the Cypher query, summarising the details of the SCADA features which contribute to the blade positioning error fault event, ultimately affecting the turbine's pitch system. Such intuitive visualisation of relevant domain-specific information can be very helpful for engineers and technicians to infer (and thus fix/avert) the anomaly-causing SCADA features in one sub-system (blades) which are indirectly contributing to a fault in a different sub-system (pitch system).

  To analyse the reasons for the transformer's valid prediction, it would be useful to visualise the self-attention weights of the model in this situation. Note that not all attention visualisations are easily comprehensible by human engineers as they represent the focus elements of the model during its prediction making process–thereby, we try to provide a perspective on the model's weights based on our domain knowledge.

  Figures 11 and 12 outline the self-attention weights for the transformer model's encoder and decoder respectively, describing the model's focus during the prediction when the source and target sequences attend to themselves. As can clearly be seen, the encoder places significant emphasis on the word *positioning* corresponding to *some*, *causes* and *turbine* etc.–which is reasonable as the question pertains to a fault event caused by a blade positioning error in the turbine. Additionally, note the other relevant attention weights such as *error* corresponding to *blade*, *blade* corresponding to *in*, *of* corresponding to *SCADA*, *examples* corresponding to *of* etc., which clearly signifies that examples of contributing SCADA features are to be retrieved for a fault which occurs in the turbine's blades. Besides, the self-attention weights at the decoder end of the model also show effective focus of the transformer on the *position* keyword corresponding to the *MATCH* fragment–which references the blade's anomaly type being in its position. Note the high attention weights for the *RETURN* keyword corresponding to multiple other Cypher query fragments such

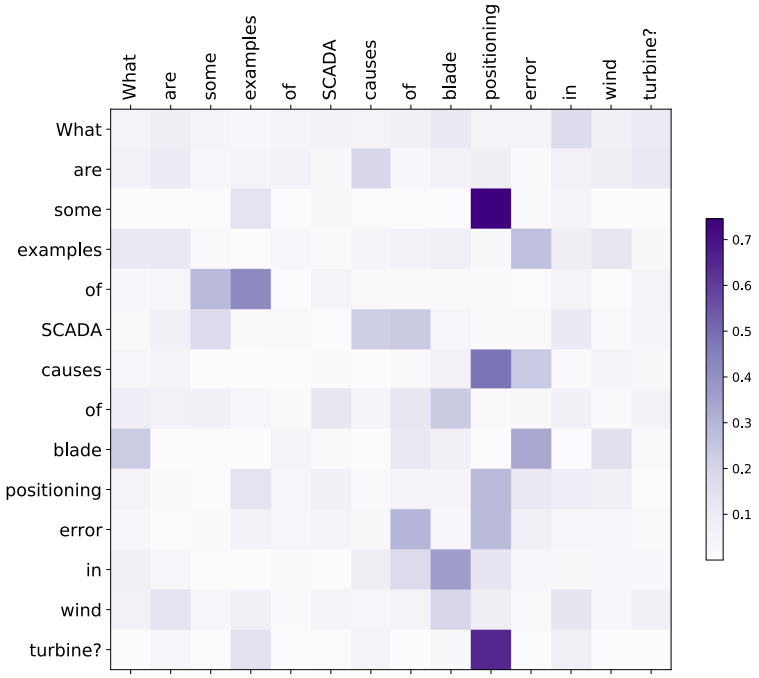


FIGURE 11. Heatmap visualisation of self-attention weights for the transformer's encoder during prediction of Cypher query in case *(vii)* of Table 8–the higher (darker) attention weights on keywords such as *positioning* corresponding to *some*, *causes* to *turbine*, *error* to *blade*, *blade* to *in*, *of* to *SCADA*, *examples* to *of* etc., highlight the model's focus on retrieving SCADA features that lead to the blade positioning fault event.

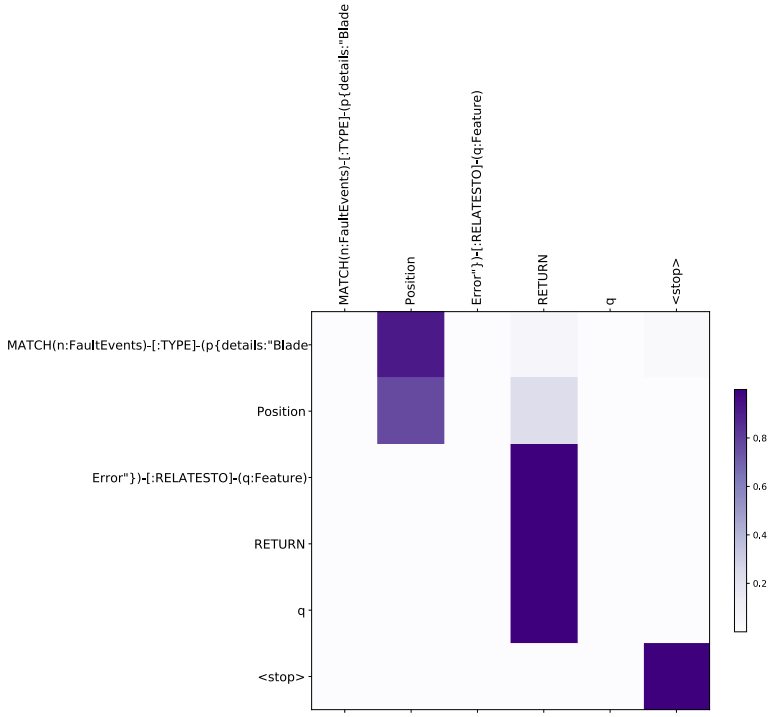
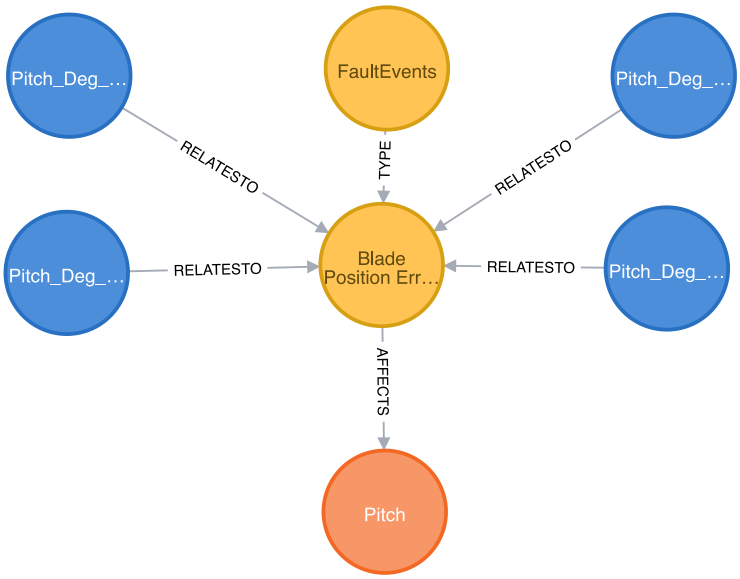


FIGURE 12. Heatmap visualisation of self-attention weights for the transformer's decoder during prediction of Cypher query in case *(vii)* of Table 8–the higher (darker) attention weights on keywords like *position* corresponding to the *MATCH* fragment references the blade's anomaly type being in its position.

as *RELATESTO* and the node label *q*–these are highly reasonable as the model aims to generate a complete valid Cypher query that can return multiple SCADA features from the KG which are relevant to a specific type of fault event (blade position error) and these thus relate to the SCADA features with the *RELATESTO*



Retrieved SCADA features in response

"q"
{"name":"Pitch_Deg_Mean","description":"Pitch Angle Mean Value","unit":"Deg","feature_no":0}
{"name":"Pitch_Deg_Min","description":"Pitch Angle Minimum Value","unit":"Deg","feature_no":1}
{"name":"Pitch_Deg_Max","description":"Pitch Angle Maximum Value","unit":"Deg","feature_no":2}
{"name":"Pitch_Deg_Stdev","description":"Pitch Angle Standard Deviation","unit":"Deg","feature_no":3}

FIGURE 13. Output on executing the Cypher query predicted by the transformer model in case *(vii)* of Table 8–the nodes shown represent the SCADA features which contribute to the blade positioning error fault event. Engineers and technicians can leverage the intuitive visualisation of nodes and properties retrieved from the KG to fix/avert potential faults in the pitch system which can occur indirectly due to anomaly-causing SCADA features in a different turbine sub-system (blades).

relationship and are referenced by the *n* node label, which the model rightly focuses on.

- For case *(viii)* of Table 8, both models fail to accurately predict the Cypher query for retrieving predictive actions required for the turbine's power cabinet. While the Seq2Seq(Att) model makes a completely incorrect prediction of the node's name–predicting maintenance actions for the *Generator* instead of the *Power Cabinet*, the transformer model still accurately infers the node's name in this scenario. However, the transformer predicts the Cypher query towards retrieving associated nodes based on the *ACTION* relationship rather than the *PredictiveActivities* property of the *Power Cabinet* node itself. This is still a relatively less fatal error, as the incorrect query generated by the Seq2Seq(Att) would predict activities for the *Generator*, which is a specific internal sub-component of the *Power Cabinet*, while the *Transformer* would return specific predictive activities for the associated events (e.g. SCADA features and alarms) that affect the *Power Cabinet* rather than the generic system itself (reflected by the system's individual property). Thus, the Cypher query generated by the transformer (despite not being completely accurate) is still realistic to avert faults by rectifying SCADA features and alarms (which directly affect the *Power Cabinet*) in this scenario.

VIII. DISCUSSION

By mapping natural language questions posed by engineers & technicians to the appropriate Cypher queries for retrieving domain-specific information, the proposed approach provides a completely automated QA system for O&M of wind turbines. The approach is also effective as whenever a Cypher query representation is correctly predicted for a natural language question, the information retrieved would always be factually correct as it is retrieved from a domain-specific KG

in the wind industry. However, there are some significant limitations of this study which are caused due the unavailability of large amounts of domain-specific O&M information openly for research & development. While we have demonstrated that techniques such as paraphrasing can support data augmentation and provide a highly effective solution for QA, there are still some key challenges which our study faces due to utilisation of the Quora Question Pairs database for fine-tuning the T5 transformer model towards data augmentation of our original domain-specific corpus in the absence of any other source of viable information. Moreover, our error and output analysis has shown that in some (rare) situations, neither the transformer nor the Seq2Seq(Att) model are able to correctly predict Cypher queries for information retrieval, which would mean that the O&M operators would still need to rely on human feedback and existing documentation (e.g. maintenance manuals).

We do not claim that our QA system can be directly utilised in the wind industry in its present state–the system would still require human engineers to analyse the responses provided and account for the incorrect decisions which the AI-based QA system can make, particularly in situations wherein the questions posed by engineers are significantly different from the training data, which, despite having significant amount of variations and diversity introduced by paraphrasing may still suffer from rarer alarm events and faults which our models have not witnessed previously. We believe that the QA system can be improved further in the future by utilising human inputs for continuously optimising the trained models and potential avenues for development of larger amounts of domain-specific O&M information by the wind farm operators–which continues to become closer to our vision with data-driven decision support becoming increasingly popular in the wind industry. In future, we plan to develop such datasets and make them publicly available to the expert and intelligent systems community, which can hopefully encourage further research in this direction to help AI-based interactive decision support systems transition from academic labs to real-world operational use-cases in the industry–particularly within the wider context of *AI for Social Good* towards tackling climate change.

IX. CONCLUSION

In this paper, we have presented a novel application of AI-based intelligent QA systems in a domain-specific real-world application towards supporting the O&M of wind turbines. We see our research as a case study in how expert systems can make interdisciplinary contributions towards pressing topics such as climate change and support a transition to renewable energy sources by making them more reliable using AI/NLP techniques. Our approach is not unique to wind energy and is transferable to any complex system that requires non-trivial decision making. By leveraging a domain-specific KG for information retrieval of O&M strategies and developing a specialised corpus of natural language questions-graph query language pairs, we have demonstrated the promising role of training AI models for automated reasoning during QA in a domain-specific application for the wind energy sector. Our proposed approach has utilised transformers for automatically generating the code for querying the KG during O&M based on natural language questions posed by engineers & technicians. Experiments with a Seq2Seq(Att) baseline model and the transformer have shown that while the transformer only marginally outperforms the Seq2Seq(Att) model in terms of accuracy, it takes one-ninth the time to train in comparison. On considering the environmental impact of our learning models and the overall goal of tackling climate change with AI, we have shown that transformers are highly promising for automated QA in the wind industry, achieving optimal performance at the lowest computational cost. We also include a brief video demonstration of the proposed QA system.[16]

Despite the promising results, our study has some significant limitations–As we utilise the Quora Questions Pair database for fine-tuning the pre-trained T5 model for performing data augmentation due to absence of any domain-specific corpus available in the wind industry, in some rare circumstances, the model can generate outputs which are completely inaccurate (or partially accurate). In these situations, it would require the domain experts to decide whether or not to consider the model's generated responses. However, we believe that this can possibly be overcome in future by optimising the model with higher quantity (and quality) of domain-specific information (such as maintenance manuals), as it becomes available in the wind industry. Additionally, reinforcement learning approaches can potentially be used in the future to continuously improve and fine-tune the QA system based on interactive human-in-the-loop feedback from engineers.

We envisage that our QA system can support O&M by helping provide appropriate O&M strategies to engineers for fixing/averting faults and operational inconsistencies and reducing the decision making time. While we do not claim that the automated QA system proposed in this paper can directly be utilised in the wind industry in its present form–we are optimistic that future work in optimising the transformer model with human feedback from domain experts and incorporating wider multimodal information beyond the single turbine which we focused on in this paper (such as maintenance records, SCADA data and alarm logs from multiple wind farms) can help the system to potentially transition to real-world and real-time deployment in operational wind turbines. We hope that our study can potentially pave the way to encouraging further research in developing AI-based QA systems for real-world industrial applications towards tackling climate change by helping make present-day energy systems smarter and more reliable.

[16]Demonstration of the QA system: `https://www.youtube.com/watch?v=Mx9SIW_7FsE`

DATA AVAILABILITY

The datasets utilised in this paper are made publicly available on GitHub at https://github.com/joyjitchatterjee/WindTurbine-QAKG.

ACKNOWLEDGMENT

The authors acknowledge the Offshore Renewable Energy Catapult (ORE Catapult), U.K., for providing their data from the Levenmouth Demonstration Turbine (LDT),[17] which helped them derive valuable domain-specific insights to develop the QA system. They also thank the Skillwind project for the publicly available Skillwind Maintenance Manual[18] which was used in developing the domain-specific KG. They are also grateful to the Department of Computer Science and Technology and the Faculty of Science and Engineering, University of Hull, U.K., for supporting their research.

REFERENCES

[1] S. K. Dwivedi and V. Singh, "Research and reviews in question answering system," *Proc. Technol.*, vol. 10, pp. 417–424, Jan. 2013.
[2] S. Park, S. Kwon, B. Kim, S. Han, H. Shim, and G. G. Lee, "Question answering system using multiple information source and open type answer merge," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Demonstrations*, 2015, pp. 111–115.
[3] R. Srihari and W. Li, "A question answering system supported by information extraction," in *Proc. 6th Conf. Appl. Natural Lang. Process.*, 2000, pp. 166–172.
[4] S. Liu, Y.-X. Zhong, and F.-J. Ren, "Interactive question answering based on FAQ," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, M. Sun, M. Zhang, D. Lin, and H. Wang, Eds. Berlin, Germany: Springer, 2013, pp. 73–84.
[5] S. Small, T. Liu, N. Shimizu, and T. Strzalkowski, "HITIQA: An interactive question answering system a preliminary report," in *Proc. ACL Workshop Multilingual Summarization Question Answering*, 2003, pp. 46–53.
[6] H. Zafar, M. Dubey, J. Lehmann, and E. Demidova, "IQA: Interactive query construction in semantic question answering systems," *J. Web Semantics*, vol. 64, Oct. 2020, Art. no. 100586.
[7] X. Xie, W. Song, L. Liu, C. Du, and H. Wang, "Research and implementation of automatic question answering system based on ontology," in *Proc. 27th Chin. Control Decis. Conf. (CCDC)*, May 2015, pp. 1366–1370.
[8] Q. Yu, W. Lam, and Z. Wang, "Responding E-commerce product questions via exploiting QA collections and reviews," in *Proc. 27th Int. Conf. Comput. Linguistics*, Santa Fe, NM, USA: Association for Computational Linguistics, Aug. 2018, pp. 2192–2203.
[9] W. A. Elnozahy, G. A. El Khayat, L. Cheniti-Belcadhi, and B. Said, "Question answering system to support university students' orientation, recruitment and retention," *Proc. Comput. Sci.*, vol. 164, pp. 56–63, Jan. 2019.
[10] S. Ou, V. Pekar, C. Orasan, C. Spurk, and M. Negri, "Development and alignment of a domain-specific ontology for question answering," in *Proc. 6th Int. Conf. Lang. Resour. Eval. (LREC)*. Marrakech, Morocco: European Language Resources Association, May 2008, pp. 1–8.
[11] J. E. Daniel, W. Brink, R. Eloff, and C. Copley, "Towards automating healthcare question answering in a noisy multilingual low-resource setting," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 948–953.
[12] T. R. Goodwin and S. M. Harabagiu, "Medical question answering for clinical decision support," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 297–306.
[13] A. Saibene, M. Assale, and M. Giltri, "Expert systems: Definitions, advantages and issues in medical field applications," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114900.

[17]Platform for Operational Data: https://pod.ore.catapult.org.uk

[18]Skillwind Maintenance Manual: https://skillwind.com/wp-content/uploads/2017/08/SKILWIND_Maintenance_1.0.pdf

[14] M. A. C. Soares and F. S. Parreiras, "A literature review on question answering techniques, paradigms and systems," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 6, pp. 635–646, Jul. 2020.
[15] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Seattle, WA, USA: Association for Computational Linguistics, Oct. 2013, pp. 1533–1544.
[16] M. Vegupatti, M. Nickles, and B. R. Chakravarthi, "Simple question answering over a domain-specific knowledge graph using BERT by transfer learning," in *Proc. 28th Irish Conf. Artif. Intell. Cogn. Sci.*, vol. 2771, L. Longo, L. Rizzo, E. Hunter, and A. Pakrashi, Eds. Dublin, Republic of Ireland: CEUR, 2020, pp. 289–300.
[17] M. Vegupatti, M. Nickles, and B. R. Chakravarthi, "Simple question answering over a domain-specific knowledge graph using BERT by transfer learning," in *Proc. AICS*, vol. 2771, L. Longo, L. Rizzo, E. Hunter, and A. Pakrashi, Eds., 2020, pp. 289–300.
[18] D. Rolnick *et al.*, "Tackling climate change with machine learning," *ACM Comput. Surv.*, vol. 55, no. 2, p. 96, Feb. 2022, doi: 10.1145/3485128.
[19] A. S. Darwish and R. Al-Dabbagh, "Wind energy state of the art: Present and future technology advancements," *Renew. Energy Environ. Sustainability*, vol. 5, p. 7, Apr. 2020.
[20] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review," *Renew. Energy*, vol. 133, pp. 620–635, Apr. 2019.
[21] S. Ozturk, V. Fthenakis, and S. Faulstich, "Failure modes, effects and criticality analysis for wind turbines considering climatic regions and comparing geared and direct drive wind turbines," *Energies*, vol. 11, no. 9, p. 2317, Sep. 2018.
[22] J. Chatterjee and N. Dethlefs, "Deep learning with knowledge transfer for explainable anomaly prediction in wind turbines," *Wind Energy*, vol. 23, no. 8, pp. 1693–1710, Apr. 2020.
[23] W. Yang and J. Jiang, "Wind turbine condition monitoring and reliability analysis by SCADA information," in *Proc. 2nd Int. Conf. Mechanic Autom. Control Eng.*, Jul. 2011, pp. 1872–1875.
[24] R. K. Ibrahim, J. Tautz-Weinert, and S. J. Watson, "Neural networks for wind turbine fault detection via current signature analysis," in *Proc. WindEurope Summit*. Hamburg, Germany, Sep. 2016, pp. 1–7.
[25] J. Lei, C. Liu, and D. Jiang, "Fault diagnosis of wind turbine based on long short-term memory networks," *Renew. Energy*, vol. 133, pp. 422–432, Apr. 2019.
[26] P. Colon-Hernandez, C. Havasi, B. Jason Alonso, M. Huggins, and C. Breazeal, "Combining pre-trained language models and structured knowledge," 2021, *arXiv:2101.12294*.
[27] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: Reasoning with language models and knowledge graphs for question answering," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2021, pp. 535–546.
[28] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," 2015, *arxiv.1506.02075*. [Online]. Available: https://arxiv.org/abs/1506.02075, doi: 10.48550/arxiv.1506.02075.
[29] X. He and D. Golub, "Character-level question answering with attention," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1598–1607.
[30] E. Quaeghebeur, S. S. Perez-Moreno, and M. B. Zaaijer, "WESgraph: A graph database for the wind farm domain," *Wind Energy Sci.*, vol. 5, no. 1, pp. 259–284, Feb. 2020.
[31] M. Lv, B. Duan, H. Jiang, and D. Dong, "Application of knowledge graph technology in unified management platform for wind power data," in *Proc. IECON 46th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2020, pp. 1762–1766.
[32] D. Küçük and Y. Arslan, "Semi-automatic construction of a domain ontology for wind energy using Wikipedia articles," *Renew. Energy*, vol. 62, pp. 484–489, Feb. 2014.
[33] E. Amador-Domínguez, E. Serrano, D. Manrique, and J. F. D. Paz, "Prediction and decision-making in intelligent environments supported by knowledge graphs, a systematic review," *Sensors*, vol. 19, no. 8, p. 1774, Apr. 2019.
[34] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "CodeBERT: A pre-trained model for programming and natural languages," in *Proc. Findings Assoc. Comput. Linguistics, (EMNLP)*, 2020, pp. 1536–1547.
[35] U. Kusupati and V. R. T. Ailavarapu, "Natural language to code using transformers," 2022, *arXiv:2202.00367*.

[36] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2017, pp. 440–450.
[37] C. Cummins, P. Petoumenos, Z. Wang, and H. Leather, "Synthesizing benchmarks for predictive modeling," in *Proc. IEEE/ACM Int. Symp. Code Gener. Optim. (CGO)*, Feb. 2017, pp. 86–99.
[38] K. Singh, I. Lytra, A. S. Radhakrishna, S. Shekarpour, M.-E. Vidal, and J. Lehmann, "No one is perfect: Analysing the performance of question answering components over the DBpedia knowledge graph," *J. Web Semantics*, vol. 65, Dec. 2020, Art. no. 100594.
[39] K. Affolter, K. Stockinger, and A. Bernstein, "A comparative survey of recent natural language interfaces for databases," *VLDB J.*, vol. 28, no. 5, pp. 793–819, Oct. 2019.
[40] S. Liang, K. Stockinger, T. M. de Farias, M. Anisimova, and M. Gil, "Querying knowledge graphs in natural language," *J. Big Data*, vol. 8, no. 1, pp. 1–23, Dec. 2021.
[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 1–11.
[42] J. Chatterjee and N. Dethlefs, "XAI4Wind: A multimodal knowledge graph database for explainable decision support in operations & maintenance of wind turbines," 2020, *arxiv.2012.10489*. [Online]. Available: https://arxiv.org/abs/2012.10489, doi: 10.48550/arxiv.2012.10489.
[43] J. Chatterjee, "The blessings of explainable AI in operations & maintenance of wind turbines," Ph.D. thesis, Dept. Comput. Sci. Technol., Univ. Hull, Hull, U.K., 2021.
[44] D. Mollá and J. L. Vicedo, "Question answering in restricted domains: An overview," *Comput. Linguistics*, vol. 33, no. 1, pp. 41–61, Mar. 2007.
[45] J. Clarke, D. Goldwasser, M.-W. Chang, and D. Roth, "Driving semantic parsing from the world's response," in *Proc. 14th Conf. Comput. Natural Lang. Learn.* Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 18–27.
[46] T. Kwiatkowksi, L. Zettlemoyer, S. Goldwater, and M. Steedman, "Inducing probabilistic CCG grammars from logical form with higher-order unification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Cambridge, MA, USA: Association for Computational Linguistics, Oct. 2010, pp. 1223–1233.
[47] M. J. Zelle and J. R. Mooney, "Learning to parse database queries using inductive logic programming," in *Proc. 13th Nat. Conf. Artif. Intell.*, vol. 2. Palo Alto, CA, USA: AAAI Press, 1996, pp. 1050–1055.
[48] M. Pavlić, Z. D. Han, and A. Jakupović, "Question answering with a conceptual framework for knowledge-based system development 'node of knowledge,'" *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5264–5286, Jul. 2015.
[49] A. Aghaebrahimian and F. Jurčíček, "Open-domain factoid question answering via knowledge graph search," in *Proc. Workshop Hum.-Comput. Question Answering*, 2016, pp. 22–28.
[50] I. Habernal and M. Konopík, "SWSNL: Semantic web search using natural language," *Expert Syst. Appl.*, vol. 40, no. 9, pp. 3649–3664, Jul. 2013.
[51] Y.-J. Han, S.-B. Park, and S.-Y. Park, "A natural language interface concordant with a knowledge base," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–15, Jan. 2016.
[52] M. K. Rohil, R. K. Rohil, D. Rohil, and A. Runthala, "Natural language interfaces to domain specific knowledge bases: An illustration for querying elements of the periodic table," in *Proc. IEEE 17th Int. Conf. Cognit. Informat. Cognit. Comput. (ICCI*CC)*, Jul. 2018, pp. 517–523.
[53] M. A. Paredes-Valverde, J. Noguera-Arnaldos, C. A. Rodríguez-Enríquez, R. Valencia-García, and G. Alor-Hernández, "A natural language interface to ontology-based knowledge bases," in *Proc. 12th Int. Conf. Distrib. Comput. Artif. Intell.*, S. Omatu, Q. M. Malluhi, S. R. Gonzalez, G. Bocewicz, E. Bucciarelli, G. Giulioni, and F. Iqba, Eds. Cham, Switzerland: Springer, 2015, pp. 3–10.
[54] P. Tong, Q. Zhang, and J. Yao, "Leveraging domain context for question answering over knowledge graph," *Data Sci. Eng.*, vol. 4, no. 4, pp. 323–335, Dec. 2019.
[55] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with freebase," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 1. Baltimore, MD, USA, Jun. 2014, pp. 956–966.
[56] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2014, pp. 1415–1425.
[57] S. Reddy, M. Lapata, and M. Steedman, "Large-scale semantic parsing without question-answer pairs," *Trans. Assoc. Comput. Linguistics*, vol. 2, pp. 377–392, Dec. 2014.
[58] Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 423–433.
[59] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," *Comput. Linguistics*, vol. 39, no. 2, pp. 389–446, Jun. 2013.
[60] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer, "Scaling semantic parsers with on-the-fly ontology matching," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Seattle, WA, USA: Association for Computational Linguistics, Oct. 2013, pp. 1545–1556.
[61] S. Mohammed, P. Shi, and J. Lin, "Strong baselines for simple question answering over knowledge graphs with and without neural networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol. (Short Papers)*, vol. 2, 2018, pp. 291–296.
[62] I. Portugal, P. Alencar, and D. Cowan, "A preliminary survey on domain-specific languages for machine learning in big data," in *Proc. IEEE Int. Conf. Softw. Sci., Technol. Eng. (SWSTE)*, Jun. 2016, pp. 108–110.
[63] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer, "Neural network-based question answering over knowledge graphs on word and character level," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1211–1220.
[64] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," in *Proc. COLING 26th Int. Conf. Comput. Linguistics, Tech. Papers*. Osaka, Japan, Dec. 2016, pp. 1746–1756.
[65] D. Lukovnikov, A. Fischer, and J. Lehmann, "Pretrained transformers for simple question answering over knowledge graphs," 2020, *arXiv:2001.11985*.
[66] J. Plepi, E. Kacupaj, K. Singh, H. Thakkar, and J. Lehmann, "Context transformer with stacked pointer networks for conversational question answering over knowledge graphs," 2021, *arxiv.2103.07766*. [Online]. Available: https://arxiv.org/abs/2103.07766, doi: 10.48550/arxiv.2103.07766.
[67] T. H. V. Phan and P. Do, "BERT+vnKG: Using deep learning and knowledge graph to improve Vietnamese question answering system," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, pp. 1–8, 2020.
[68] E. Kacupaj, J. Plepi, K. Singh, H. Thakkar, J. Lehmann, and M. Maleshkova, "Conversational question answering over knowledge graphs with transformer and graph attention networks," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics, Main Volume*, 2021, pp. 850–862.
[69] S. Haiduc, J. Aponte, L. Moreno, and A. Marcus, "On the use of automated text summarization techniques for summarizing source code," in *Proc. 17th Work. Conf. Reverse Eng.*, Oct. 2010, pp. 35–44.
[70] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, "Summarizing source code using a neural attention model," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2016, pp. 2073–2083.
[71] X. Hu, Y. Wei, G. Li, and Z. Jin, "CodeSum: Translate program language to natural language," 2017, *arXiv:1708.01837*.
[72] M. Allamanis, H. Peng, and A. C. Sutton, "A convolutional attention network for extreme summarization of source code," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*. New York, NY, USA, Mar. 2016, pp. 2091–2100.
[73] W. Ling, P. Blunsom, E. Grefenstette, K. M. Hermann, T. Kočiský, F. Wang, and A. Senior, "Latent predictor networks for code generation," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2016, pp. 599–609.
[74] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 731–742.
[75] P. Hohenecker and T. Lukasiewicz, "Ontology reasoning with deep neural networks," *J. Artif. Intell. Res.*, vol. 68, pp. 503–540, Jul. 2020.
[76] P. Vougiouklis, E. Maddalena, J. Hare, and E. Simperl, "Point at the triple: Generation of text summaries from knowledge base triples," *J. Artif. Intell. Res.*, vol. 69, pp. 1–31, Sep. 2020.
[77] H. Chockler, P. Kesseli, D. Kroening, and O. Strichman, "Learning the language of software errors," *J. Artif. Intell. Res.*, vol. 67, pp. 881–903, Apr. 2020.
[78] G. J. D. R. Hains, Y. Khmelevsky, and T. Tachon, "From natural language to graph queries," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.

[79] C. Alberti, D. Andor, E. Pitler, J. Devlin, and M. Collins, "Synthetic QA corpora generation with roundtrip consistency," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 6168–6173.
[80] A. Sadid Hasan, B. Liu, J. Liu, A. Qadir, K. Lee, V. Datla, A. Prakash, and O. Farri, "Neural clinical paraphrase generation with attention," in *Proc. Clin. Natural Lang. Process. Workshop (ClinicalNLP)*. Osaka, Japan, Dec. 2016, pp. 42–53.
[81] L. Dong, J. Mallinson, S. Reddy, and M. Lapata, "Learning to paraphrase for question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 875–886.
[82] A. Prakash, A. S. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural paraphrase generation with stacked residual LSTM networks," in *Proc. COLING 26th Int. Conf. Comput. Linguistics, Tech. Papers*. Osaka, Japan, Dec. 2016, pp. 2923–2934.
[83] E. Kacupaj, B. Banerjee, K. Singh, and J. Lehmann, "ParaQA: A question answering dataset with paraphrase responses for single-turn conversation," 2021, *arxiv.2103.07771*. [Online]. Available: https://arxiv.org/abs/2103.07771, doi: 10.48550/arxiv.2103.07771.
[84] Z. Tan, S. Wang, Z. Yang, G. Chen, X. Huang, M. Sun, and Y. Liu, "Neural machine translation: A review of methods, resources, and tools," *AI Open*, vol. 1, pp. 5–21, Jan. 2020.
[85] F. M. S. López and E. S. D. L. Cruz, "Literature review about Neo4j graph database as a feasible alternative for replacing RDBMS," *Ind. Data*, vol. 18, no. 2, pp. 135–139, 2015.
[86] W. C. Gan and H. T. Ng, "Improving the robustness of question answering systems to question paraphrasing," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 6065–6075.
[87] Y. Fu, Y. Feng, and J. P. Cunningham, "Paraphrase generation with latent bag of words," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–12.
[88] K. R. McKeown, "Paraphrasing questions using given and new information," *Comput. Linguistics*, vol. 9, no. 1, pp. 1–10, 1983.
[89] E. Egonmwan and Y. Chali, "Transformer and seq2seq model for paraphrase generation," in *Proc. 3rd Workshop Neural Gener. Transl.*, 2019, pp. 249–255.
[90] Z. Li, X. Jiang, L. Shang, and H. Li, "Paraphrase generation with deep reinforcement learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3865–3878.
[91] L. Qian, L. Qiu, W. Zhang, X. Jiang, and Y. Yu, "Exploring diverse expressions for paraphrase generation," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 3173–3182.
[92] F. Rinaldi, J. Dowdall, K. Kaljurand, M. Hess, and D. Mollá, "Exploiting paraphrases in a question answering system," in *Proc. 2nd Int. Workshop Paraphrasing*, 2003, pp. 25–32.
[93] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[94] C. Fellbaum, *WordNet*. Hoboken, NJ, USA: Wiley, 2012.
[95] T. Niu, S. Yavuz, Y. Zhou, N. S. Keskar, H. Wang, and C. Xiong, "Unsupervised paraphrasing with pretrained language models," 2020, *arxiv.2010.12885*. [Online]. Available: https://arxiv.org/abs/2010.12885, doi: 10.48550/arxiv.2010.12885.
[96] S. Witteveen and M. Andrews, "Paraphrasing with large language models," in *Proc. 3rd Workshop Neural Gener. Transl.*, 2019, pp. 215–220.
[97] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
[98] L. Sharma, L. Graesser, N. Nangia, and U. Evci, "Natural language understanding with the Quora question pairs dataset," 2019, *arXiv:1907.01041*.
[99] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.
[100] J. J. Bird, A. Ekárt, and D. R. Faria, "Chatbot interaction with artificial intelligence: Human data augmentation with T5 and language transformer ensemble for text classification," 2020, *arxiv.2010.05990*. [Online]. Available: https://arxiv.org/abs/2010.05990, doi: 10.48550/arxiv.2010.05990.
[101] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python*. Sebastopol, CA, USA: O'Reilly Media, 2009.
[102] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.
[103] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Bellevue, WA, USA, Jun. 2011, pp. 1017–1024.
[104] N. Dethlefs, "Domain transfer for deep natural language generation from abstract meaning representations," *IEEE Comput. Intell. Mag.*, vol. 12, no. 3, pp. 18–28, Aug. 2017.
[105] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," 2017, *arXiv:1702.05659*.
[106] L. Wu, F. Tian, Y. Xia, Y. Fan, T. Qin, L. Jian-Huang, and T.-Y. Liu, "Learning to teach with dynamic loss functions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 6467–6478.
[107] R. van der Meer, C. W. Oosterlee, and A. Borovykh, "Optimally weighted loss functions for solving PDEs with neural networks," *J. Comput. Appl. Math.*, vol. 405, May 2022, Art. no. 113887.
[108] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1–11.
[109] S. Takahashi and K. Tanaka-Ishii, "Cross entropy of neural language models at infinity—A new bound of the entropy rate," *Entropy*, vol. 20, no. 11, p. 839, Nov. 2018.
[110] Z. Xie, Y. Huang, Y. Zhu, L. Jin, Y. Liu, and L. Xie, "Aggregation cross-entropy for sequence recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6538–6547.
[111] C. Sammut and G. I. Webb, Eds., *Mean Squared Error*. Boston, MA, USA: Springer, 2010, p. 653.
[112] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[113] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Oct. 2014, pp. 1724–1734.
[114] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Proc. NIPS*, 2014, pp. 1–9.
[115] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Represent. (ICLR)*. San Diego, CA, USA, 2015, pp. 1–15.
[116] T. Wolf, "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*. Stroudsburg, PA, USA: Association for Computational Linguistics, Oct. 2020, pp. 38–45.
[117] J. Chatterjee and N. Dethlefs, "A dual transformer model for intelligent decision support for maintenance of wind turbines," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–10.
[118] M. Abadi, A. Agarwal, and P. Barham. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: tensorflow.org
[119] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Mar. 2021, pp. 610–623.
[120] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni, "Green AI," *Commun. ACM*, vol. 63, no. 12, pp. 54–63, Dec. 2020.
[121] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 248, pp. 1–43, 2020.

JOYJIT CHATTERJEE (Member, IEEE) received the bachelor's degree (Hons.) in electronics and communication engineering from Amity University, India, the Postgraduate Diploma degree in research training, and the Ph.D. degree in computer science from the University of Hull, U.K.

He was a Visiting Researcher at the University of Bremen, Germany, and Tamkang University, Taiwan, on government funded research projects. He is currently a Data Scientist (KTP Research Associate) at the University of Hull and Reckitt, U.K. He was named in the Forbes 30 Under 30 Europe list (Manufacturing and Industry) in 2022 for

his impactful work on developing AI products that can help bolster manufacturing and energy processes. At the University of Hull, his research dealt with tackling climate change with AI, by helping make wind energy sources more reliable, through explainable and intelligent decision support in their operations and maintenance. As a Global Researcher, he has coauthored articles and filed patents with reputed academicians from across Europe, North America and Asia, and delivered talks at leading research institutions globally. He has published an array of papers in computer science and engineering, including in leading A*/A ranked AI conferences/workshops, such as NeurIPS and KDD, and reputed journals, being a top-cited author. His research has received significant press/media exposure. His research interests include span deep learning, natural language generation, data analytics, knowledge graphs, causal inference, and time-series analysis. He is a member of IEEE U.K. and Ireland Section's Diversity, Equity and Inclusion Committee. He has received the Green Talents Award from the German Federal Ministry of Education and Research for his Ph.D. research and its high-potential in facilitating sustainable development. He received Professional Certificate in AI from University of Oxford, U.K. He regularly serves on the program committee/as a reviewer for leading international conferences/workshops, such as ICLR and NeurIPS, and high-impact journals.

NINA DETHLEFS received the Ph.D. degree in computational linguistics from the University of Bremen, Germany.

She has over ten years of experience in academia in research, teaching, undergraduate and postgraduate supervision, and working with external partners to translate research into practice. She is currently a Senior Lecturer in computer science at the University of Hull, U.K., where she leads and conducts research into natural language processing, applied machine learning, and wider artificial intelligence. She is also the Director of Research for computer science and technology and the Aura CDT Theme Lead for "Big data, sensors and digitalisation for the offshore environment." Before coming to Hull, she was a Research Fellow with The Interaction Laboratory, Heriot-Watt University, Edinburgh. She is also the Founding Head of the Big Data Analytics Research Group and the Admissions Lead on the EPSRC-NERC Centre for Doctoral Training in offshore wind and the environment. She is interested in the digitalisation of the offshore wind industry to make wind turbines more reliable. She is also interested in the effects of human activity on water quality and the forecasting of natural events such as floods. A central theme of her research is how information can be extracted from disparate data sources, including text, images, numeric and machine data, and social media, and presented in a human-accessible, intuitive way that is reliable, ethical, and data efficient. Her research interests include data-to-text generation, text mining, (spoken and multimodal) dialogue systems, sentiment analysis, and social media analysis. She has served on many scientific committees for leading conferences and journals and has been the Area Chair for ACL and COLING.