A Dual Transformer Model for Intelligent Decision Support for Maintenance of Wind Turbines

1st Joyjit Chatterjee
Dept. of Computer Science & Technology
University of Hull
Hull, United Kingdom
j.chatterjee-2018@hull.ac.uk

2nd Nina Dethlefs
Dept. of Computer Science & Technology
University of Hull
Hull, United Kingdom
n.dethlefs@hull.ac.uk

***Abstract*—Wind energy is one of the fastest-growing sustainable energy sources in the world but relies crucially on efficient and effective operations and maintenance to generate sufficient amounts of energy and reduce downtime of wind turbines and associated costs. Machine learning has been applied to fault prediction in wind turbines, but these predictions have not been supported with suggestions on how to avert and fix faults. We present a data-to-text generation system utilising transformers for generating corrective maintenance strategies for faults using SCADA data capturing the operational status of turbines. We achieve this in two stages: a first stage identifies faults based on SCADA input features and their relevance. A second stage performs content selection for the language generation task and creates maintenance strategies based on phrase-based natural language templates. Experiments show that our dual transformer model achieves an accuracy of up to 96.75% for alarm prediction and up to 75.35% for its choice of maintenance strategies during content-selection. A qualitative analysis shows that our generated maintenance strategies are promising. We make our human-authored maintenance templates publicly available, and include a brief video explaining our approach.**

***Index Terms*—SCADA,Transformer, Alarm Message Generation,Content Selection**

I. Introduction

As the world is moving towards sustainable energy sources, rapid technological developments have contributed to making wind energy the strongest and fastest growing renewable energy resource in recent times [1]. Wind turbines consists of an array of complex mechanical and electrical components, which lead them to experience inconsistent operational behaviour from time to time. With proper operations and maintenance (O&M) strategies, any incipient faults can be averted by helping engineers and technicians to make appropriate timely decisions. This can help save up to 30% on operational costs associated with regular maintenance of turbines [2].

While there is a growing interest in the wind industry to use machine learning to predict anomalies in O&M directly from Supervisory Control & Acquisition (SCADA) data [3], [4], there is a clear paucity of intelligent decision support systems which not only predict the occurrence of an impending fault but also generate a clear human-intelligible diagnosis of its cause(s). Further, predicting a fault is of little use unless an effective maintenance action can be proposed to avert and fix it [5].

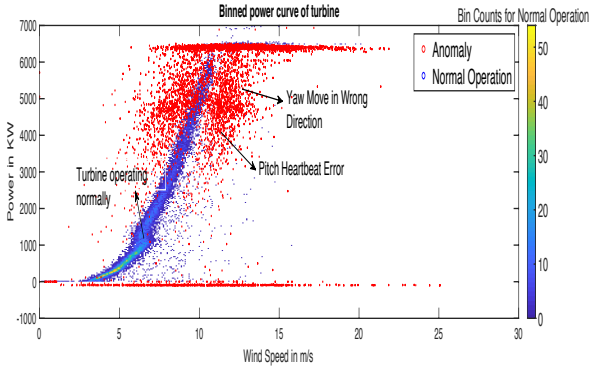

Fig. 1. Power curve outlining normal operation and anomaly. We aim to generate alarm messages and maintenance actions for the latter, taking into account the discovered causes of alarm.

In this paper, we explore the feasibility of applying natural language generation to O&M in wind turbines and generating informative alarm messages and maintenance strategies directly from the turbine's SCADA data. Figure 1 depicts the binned power curve of a wind turbine [6], which lies at the core of our generation task. For any data point that falls off the normal operation curve, we aim to **(1)** generate an alarm message highlighting the occurring fault and affected subcomponent(s) and **(2)** propose maintenance actions to fix (or avert) the fault. As our task relies on long input sequences of continuous values, and very different combinations of these can be relevant given the nature of a fault, we propose to model our task based on a transformer for sequence-to-sequence generation. Earlier work has shown [7] that a transformer's multi-head attention and principle to compute attention weights over sequences in a single iteration can make the model faster on long sequences as well as more accurate in some cases as outputs do not depend on the order of input processing [7].

We decompose our generation task into 2 stages (alarm and maintenance), utilising a transformer model at each stage. Transformer 1 takes as input a set of SCADA features capturing the operational status of the turbine, and outputs the corresponding alarm message. We also utilise the multi-head attention mechanism of Transformer 1 to obtain the relevant features which are most-likely causing the corresponding

Preprint of paper published at IJCNN 2020. J. Chatterjee and N. Dethlefs, "A Dual Transformer Model for Intelligent Decision Support for Maintenance of Wind Turbines," 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom, 2020, pp. 1-10, doi: 10.1109/IJCNN48605.2020.9206839. Final published version is at https://ieeexplore.ieee.org/document/9206839

alarm. In the second stage, Transformer 2 takes as input the sequence of features likely causing the fault along with the alarm type (i.e. the outputs of Transformer 1), and finally generates the most-effective maintenance strategy, in the form of a sequence of phrase-based natural language templates, trained from samples authored by a human domain expert.

We compare our Transformer 1 for alarm messages against conventional LSTM-based Seq2Seq models with and without attention. We choose Luong attention for the latter [8]. An evaluation in terms of percentage of alarms predicted correctly shows that the transformer network outperforms baselines (by up to 18.7%) while achieving higher computational speed of up to 28.78% compared to an encoder-decoder model with attention. We find particularly that the feature importances derived based on attention scores from the transformer are more reasonable and relevant compared to our baselines. For Transformer 2 generating maintenance strategies, we find that our model achieves an accuracy of up to 75.35%, outperforming a Seq2Seq(Att) model by up to 18.22%.

II. Related Work

Data-to-text generation has been explored for a multitude of domains including weather forecast generation [9], spatial navigation [10], sports commentaries [11] or knowledge entities [12], amongst others. Dialogue systems also often contain an NLG component to transform a semantic input form into an utterance that can be presented to a user [13], [14].

A. Deep learning models for NLG

Much recent work in statistical NLG has been based on sequence-to-sequence recurrent neural networks [15] that learn a mapping between a non-linguistic input representation, e.g. a set of semantic slots to be expressed, meteorological data, pixels in an image etc., onto a sequence of words that describe the input in natural language. Several extensions have been proposed to this basic model, e.g. to semantically condition outputs to increase semantic accuracy [16], pre-process the input to increase semantic output quality [17], apply copy actions [18] or pointer sentinel networks [19] to increase the accuracy of slot transfer, or to inject linguistic information into a purely learnt model in order to capture more of the lexico-syntactic regularities of a domain [20], [21].

An alternative model to the sequence-to-sequence architecture are variational auto-encoders (VAEs) which are normally used to encode an input representation and then decode it back to its original with minimal loss. Recent models have shown that when systematically inserting "noise" into inputs, such as information on target semantic slots, VAEs can learn to transform a semantic input sequence into a lexico-syntactic surface form [22] much like other architectures. [23] show that combining VAE generation with convolutional and deconvolutional techniques helps to reduce inconsistent outputs.

B. Transformers for NLG

The transformer is a more recent model that has been shown to outperform sequence-to-sequence models on a number of tasks, including machine translation [7], natural language understanding, question answering and common sense inference [24], and video captioning [25]. The main idea is to remove the requirement of a recurrent neural network and compute attention weights over an input sequence with positional embeddings instead. The model extends conventional attention mechanisms (i.e., the output sequence attending to the input sequence) with self-attention, where both inputs and outputs attend to themselves in addition to the target attending to the source [7]. This has been shown to be more effective in some domains than the standard attention model, while the removal of recurrence reduces computational cost, potentially making transformers an attractive model for real-world and real-time data-to-text generation.

There has been limited research in the area of using transformers for data to text generation, as this still remains an under explored research area. A notable exception is [18]'s entry to the 2018 E2E challenge. The authors apply a transformer to text generation (restaurant recommendations) from discrete semantic slots and achieve good results, even though their transformer model was not their best model. [25] use transformers for video caption generation and find it to outperform multiple competitive LSTM baselines. Other related work on transformers for NLG includes [26] who train an NLG system for a new domain based on a small amount of labelled data and an extensive pre-trained language model to inherit general-purpose linguistic knowledge. While their basic model is an LSTM sequence-to-sequence architecture with a field-gating encoder (see [27]), the pre-trained language model is a 12-layer transformer [28].

We extend previous work on using transformers for NLG to a data-to-text generation scenario from continuous numeric inputs. In particular, we are interested in the quality of the transformer's attention scores towards providing transparent decision making beyond accurate predictions.

III. Learning model

A. The Transformer model

We model our NLG system as an architecture involving 2 stages: **Stage (1)** takes as input a sequence of SCADA features and outputs an alarm event description, and **Stage (2)** predicts the most effective maintenance strategy in natural language. For Stage 1, as the SCADA data from a wind turbine is a typical time-series with multiple features measured from various sensors (at generally 10 minutes intervals), the model will learn to map a sequence of features to a corresponding alarm message describing operational behaviour of the turbine at any given interval. Stage 2 can be seen as a sequence generation task as maintenance strategies can consist of multiple actions that can occur in different orders to fix or avert a fault.

The transformer model is a relatively recent architecture for sequence transduction proposed by [7], which has been shown to achieve or exceed state of the art performance on a number of sequence-to-sequence tasks, see Section II. In contrast to earlier Seq2Seq models, transformers do not rely on RNNs for

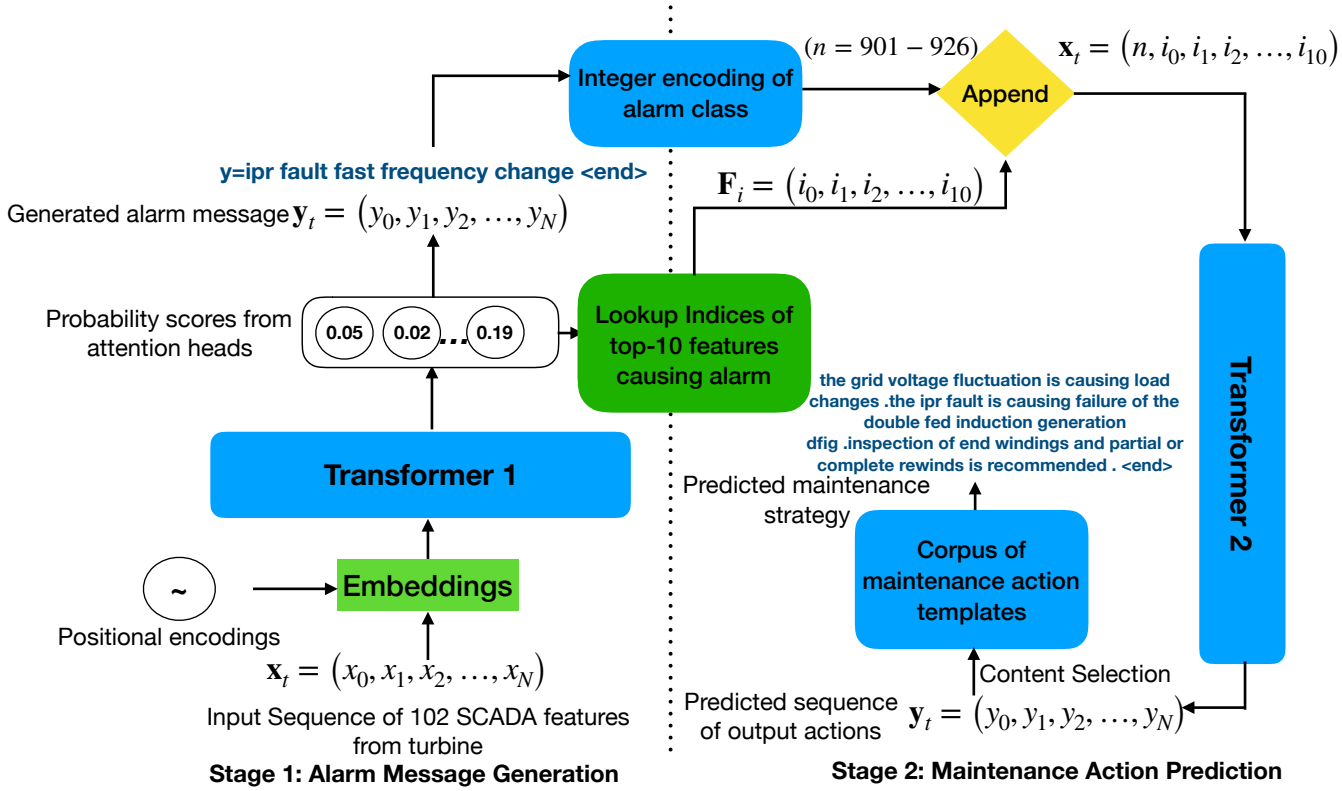

Fig. 2. Bringing together Transformer 1 for alarm generation and Transformer 2 for content selection

Time Stamp	Feature 1 (X_0)	Feature 2 (X_1).....	Feature n (X_n)	Event Description
dd/mm/yyyy hh:mm:ss	2.104	0.890	8.124	Turbine Operating Normally
dd/mm/yyyy hh:mm:ss	1.245	3.753	9.509	Pitch System Fatal Error

TABLE I
EXAMPLE OF AN INPUT DATA STRUCTURE AND CORRESPONDING ALARM LOG.

sequence processing, which gives them a number of advantages including computational efficiency. The fact that RNNs read sequences one word at a time as well as their sequential processing nature makes them less straightforward for parallel GPU processing than transformers, which rely predominantly on feed-forward networks and matrix multiplication [29].

Similar to Seq2Seq models, the transformer architecture consists of an encoder and a decoder. However, it contains multiple attention constraints, referred to as multi-head attention, which represent the essence of its performance [7], [30]. In particular, encoder and decoder apply self-attention wherein source and target sequences attend to themselves, respectively. In addition, the target will attend to the source sequence as in conventional attention architectures [15].

Figure 3 illustrates an example of visualisation of individual neurons within the query and key vectors used for computing the final attention scores for the generated alarm message in the transformer which we developed using BertViz [31]. The final output message is generated during the decoding stage, utilising past and final hidden representations received from the encoder. A softmax layer is used to capture the scores of words to be predicted in the target sequence, which in our case is the predicted alarm message. The interested reader is referred to [7] for further details on the transformer architecture.

Below, we describe the role of the transformer in each stage for predicting faults and selecting maintenance actions. We use transformer networks in both stages of our NLG architecture, for fault identification and alarms (Stage 1) as well as content selection for maintenance strategies (Stage 2).

B. Stage 1: Generating alarm messages

The first stage takes as input a sequence of SCADA features $\mathbf{x}_t = (x_0, x_1, x_2, \ldots, x_N)$ and outputs a sequence of symbols describing the internal state of the turbine $\mathbf{y}_t = (y_0, y_1, y_2, \ldots, y_N)$, where t is a time step. This is shown in Figure 2. The transformer's task in this case is to learn when a fault is likely to occur (so that an early alarm can be raised) as well as estimate likely causes of the fault. The latter we take from the network's attention weights, see Figure 7 for an example.

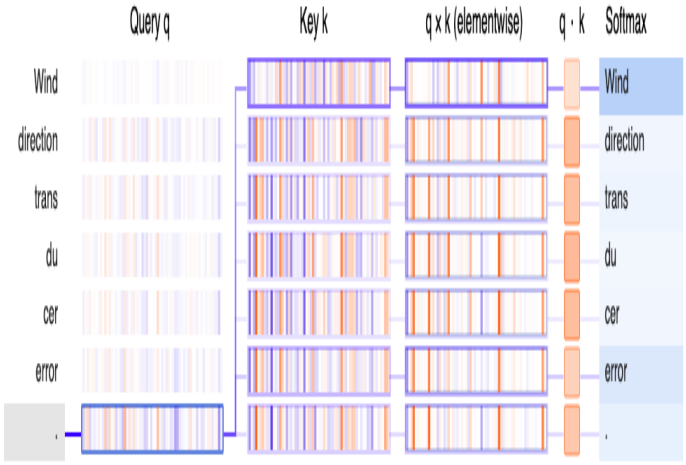

Fig. 3. Example of neuron view within the query and key vectors.

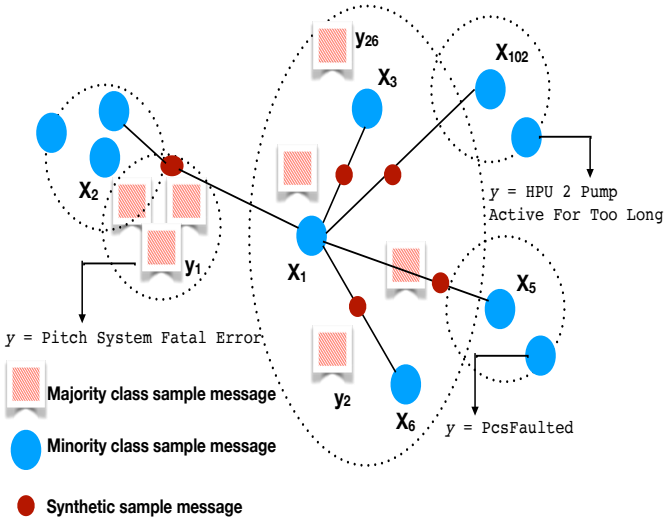

Fig. 4. Illustration of synthetic minority oversampling for wind turbine alarm messages

The predicted alarm and relevant features are passed on as input to Stage 2 for content selection and generation of maintenance strategies (to fix or avert the fault).

C. Stage 2: Content selection of maintenance actions

We model our second stage for content selection and maintenance strategy generation as a separate transformer network. Transformer 2 works in exactly the same manner as Transformer 1, and will predict a sequence of maintenance actions (to form a strategy) from an input sequence of an alarm and features that gave rise to this alarm. Figure 2 shows an example of this.

Specifically, given a generated alarm message from Transformer 1, we first integer-encode it into one of 26 alarm categories ($n = 901-926$). Then, the alarm class of generated message n is appended with a list of the top-10 features $\mathbf{F}_i = (i_0, i_1, i_2, \ldots, i_{10})$ causing the alarm obtained with the attention mechanism of Transformer 1 to obtain a new sequence of features $\mathbf{x}_t = (n, i_0, i_1, i_2, \ldots, i_{10})$. This serves as an input to Transformer 2, and finally, it predicts the sequence of most-effective maintenance actions by selecting relevant templates from a dictionary of key-value pairs of available maintenance actions. Please refer to Section V for more details on the phrase-based natural language templates used in Stage 2 of our transformer. Figure 2 outlines the topology of the combination of Transformers 1&2 used for predicting maintenance actions.

IV. Data Description and Preprocessing

For our study, we used SCADA data measured at 10-minute intervals from the Levenmouth Demonstration Turbine (LDT)[1], a 7 MW rated operational offshore wind turbine in Scotland. We used 102 features from the LDT logs as an input sequence to Transformer 1 to predict the corresponding alarm message (referred to as event description in the original data). See Table I for an example of a data structure and error description, where features 1-n are sensor readings.

These event descriptions were originally prepared by human experts and our goal is to predict an alarm type and message from the 102 operational features. In total, the data contains 26 discrete classes of alarms with corresponding human-written messages. The 102 features used contain various metrics, ranging from several electrical, temperature, pressure readings etc. captured by operational sensors as well as meteorological data such as humidity, wind conditions, etc.

a) Addressing class imbalance: A challenge with our dataset is a substantial class imbalance across alarm types as some occur much more frequently than others, thus leading to an imbalanced dataset of generation examples. For instance, the *Pitch System Fatal Error* alarm accounted for 5,050 cases owing to the fairly common pitch angle disorientation in turbines, while *HPU 2 Pump Active For Too Long* only accounted for 2,525 cases, exactly half of the former as these are rarer messages. Even rarer cases include *PcsFaulted* for instance, with only 101 cases. Training over the dataset as is can lead to a model learning a biased generation policy that would prefer the frequent examples in most cases and not learn a valid representation of some of the rarer occurrences.

One way to address this situation is to generate more data in simulation and from heuristics. In our case, this was deemed too costly as we would require access to engineers to help us annotate additional alarm messages for various conditions. An alternative is to reproduce data points, e.g. including random noise and using uniform random sampling [32]. Such techniques are simple and can be effective but contain a risk of changing the overall data distribution.

We finally decide to use the Synthetic Minority Oversampling Technique (SMOTE) [33], a popular statistical technique

[1]Platform for Operational Data (POD) Disseminated by ORE Catapult: https://pod.ore.catapult.org.uk

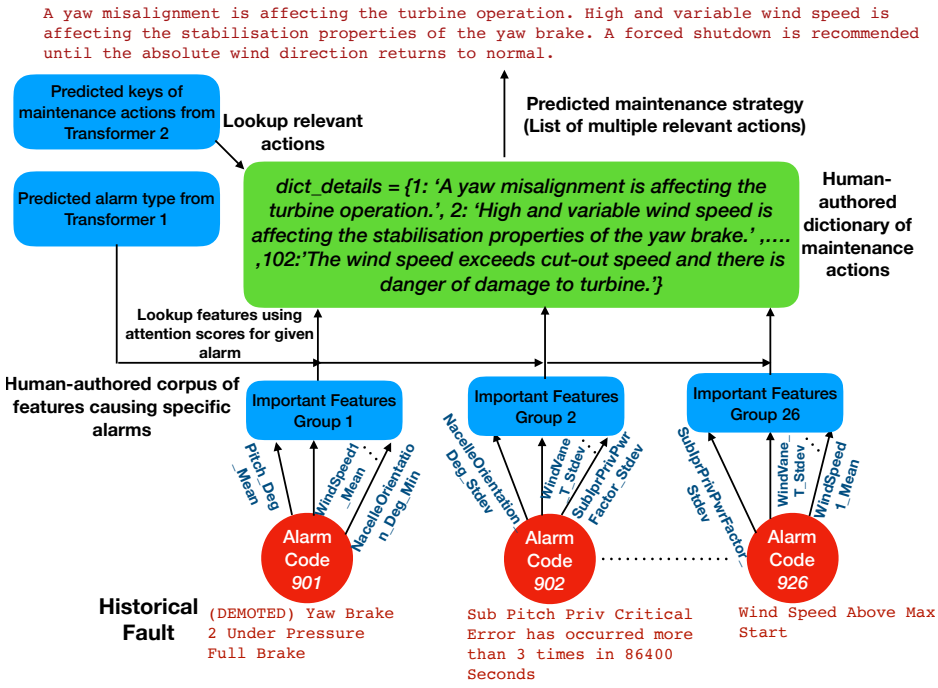

Fig. 5. Development of templates for maintenance actions

that addresses class imbalance by generating more samples of a certain type, introducing noise and variation but at the same time preserving the original data distribution. In particular, the idea is to inspect successive samples of a given training set and calculating the distance between the k-nearest neighbours across the feature space. The new synthetic data points are generated by multiplying the vectorial distance between the nearest neighbours across the original dataset with a random real number between 0 and 1, and summing it with the present value of the sample in the feature space. Figure 4 illustrates this for our dataset. Considering the 102 features in our dataset ranging from $\mathbf{X}_t = (X_1, \ldots, X_{102})$, SMOTE balances the samples for the minority class targets belonging to 26 different categories of alarm messages from $\mathbf{y} = (y_1, \ldots, y_{26})$ by creating a feature space mapping from the majority samples to the minority samples based on the vectorial distance between these points. We implement SMOTE using Python's *imbalanced-learn* library [34].

b) Integer encoding: Ultimately, we obtained 500 examples for each alarm type leading to an overall dataset of 13,000 samples, each with 102 SCADA input features and an alarm type output. Input sequences from a continuous range were centred and scaled in the range of 0 to 10 to facilitate unbiased predictions. To do this, we created a mapping to convert floating point values to integers by rounding the scaled values to the nearest integers.

V. Templates for Maintenance Actions

To produce the final output for our learning model, we created a human-authored corpus of maintenance actions corresponding to various faults which had historically occurred in the turbine. Given that our dataset consisted of a total of 26 discrete classes of alarms, a human domain-expert created a total of 167 sub-phrases spanning these classes, where each sub-phrase details a specific maintenance action, and a maintenance strategy (as we aim to generate) is made up of a sequence of such actions, i.e. a sequence of sub-phrases. Different alarm types can sometimes share the same maintenance strategy (if the same actions will fix a fault), or conversely, it is possible that the same alarm message will require different sets of maintenance actions (if the underlying causes for the alarms differ). Figure 6 outlines this through an example Circos visualisation of the associations between alarm classes, which we developed using [35]. As can be visualised, multiple alarm classes are not independent in terms of maintenance strategies for fixing the faults, and there is intersection amongst the actions for many cases. For instance, to fix an alarm in the yaw brake (type 1), causal association exists with maintenance action for alarm in the blades (type 19). Therefore, as two different alarms can share the same maintenance actions, of these 167 templates, 102 sub-phrases were completely unique to any of the classes. We make our human-authored maintenance templates publicly available. [2] Template sub-phrases average to 6.073 words.

A complete maintenance strategy in this study is a text output, consisting of a selection of multiple maintenance action sub-phrases selected by the second stage of our Transformer.

[2]Turbine Maintenance Templates: https://github.com/joyjitchatterjee/TurbineMaintenanceTemplates

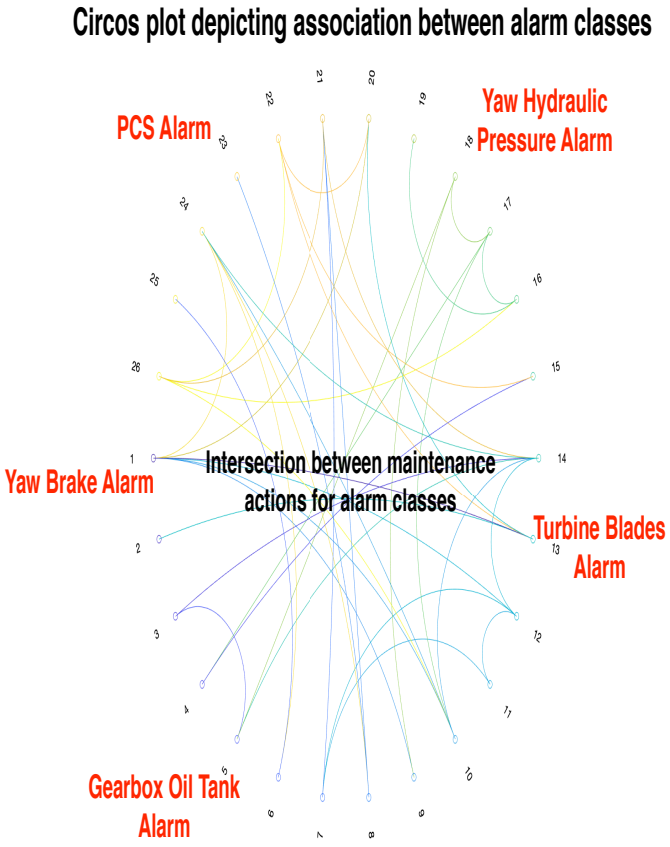

Fig. 6. Circos plot visualisation of alarm classes, for which maintenance actions are finally predicted. As can be seen, alarms are not independent, and can thus share maintenance actions.

To overcome the potential problem of training our Stage 2 model with a small corpus and class imbalance (as some maintenance actions are more prominent than others), we over-sampled the human-authored corpus of maintenance actions for given SCADA alarms and relevant features, similar to the procedure used in Section IV, and obtained an overall dataset of 1,055 samples.

Templates of sub-phrases for maintenance actions were collected and stored in the form of key-value pairs in a dictionary, see Figure 5. Given a set of indices of top-10 likely features identified by Transformer 1, and the final content-selection performed by Transformer 2, based on the human-authored maintenance action sub-phrases, our model can output an alarm type and a corresponding suggested maintenance strategy to avert or fix it.

VI. Experiments

A. Stage 1: Alarm generation

As the first stage of our dual-transformer model involves identifying the specific type of fault which occurred in the turbine, before any relevant maintenance action can be predicted, we use the SCADA sequence of input features and historically labelled alarm messages processed in Section IV for training the model. We used Tensorflow [36] for all three learning models we wish to compare:

- **Seq2Seq**: The Seq2Seq model with LSTM, see [15] for details. We use 200-dimensional word embeddings, 64 hidden neurons, a learning rate of 0.001, dropout of 0.1, Adam optimisation, and 200 training epochs.
- **Seq2Seq (Att)**: The Seq2Seq model with LSTM and Luong attention, see [37] for details, with the same hyperparameters as Seq2Seq. For attention, we use the *concat* score function to compute the alignment vectors, alongside the *dot* and *general* which Luong attention already provides.
- **Transformer**: the Transformer model with multi-head attention, see Section III-A. We use 8 multi-head attention heads, model size of 64, and 3 dense layers for each head. The learning rate was decayed based on the *WarmupThenDecaySchedule* class in Tensorflow.

We split our dataset in a 80%-20% ratio into training and test data and use a batch size of 32.

B. Stage 2: Maintenance strategy generation

To train our Stage 2 of the transformer, we learn a mapping from an input sequence of alarm type and relevant features to an output sequence of maintenance actions. The latter are represented as integers that point to a dictionary of text-based templates. We use a model size of 128 and vocabulary size of 1,300 words for this stage. We again used a 80%-20% train-test split for Stage 2 with a batch size of 16. The Stage 2 model was trained over 500 epochs.

We utilised the Seq2Seq(Att) model as a baseline for Stage 2 with LSTM and Luong attention, 128 hidden neurons, learning rate of 0.001, dropout of 0.1, Adam optimisation, and trained over 500 epochs. It was not feasible to use a vanilla Seq2Seq model without attention as a baseline at this stage, as the important features are required by our model to select appropriate content, and this can only be accomplished with an attention mechanism.

VII. Results

A. Stage 1 evaluation

Table II shows results from our objective evaluation for Stage 1 in terms of percentage of alarm types correctly predicted, performance metrics in terms of average precision, recall and F1 score as well as the computation time.

1) Objective Evaluation: In terms of percentage of alarm types correctly predicted, we can see that the **Transformer** clearly outperforms the other two models. Also, the transformer model attains the highest F1 score of up to 0.978 compared to the other baseline models, which perform very similarly with the standard **Seq2Seq** model scoring slightly higher. This is likely due to our sequences being relatively short – 5.49 symbols on average with a maximum sequence length of 14 symbols in the training data and minimal sequence length of 1 symbols. It is likely that an attention mechanism is not required to learn good representations for the shorter sequences.

Stage 1 Model	Percentage of Alarms Correctly Predicted	Computation time	Avg. Precision	Avg. Recall	Avg. F1 Score
Seq2Seq	78.054%	**1.42 min**	0.847	0.853	0.850
Seq2Seq (Att)	79.25%	4.25 min	0.862	0.853	0.857
Transformer	**96.75%**	3.30 min	**0.967**	**0.99**	**0.978**

TABLE II
STAGE 1 RESULTS IN TERMS OF PERFORMANCE METRICS FOR PREDICTED ALARM TYPES AND COMPUTATION TIME. THE BEST PERFORMING MODEL IS SHOWN IN BOLD-FACE.

We also note that (unsurprisingly) the **Seq2Seq** model is the fastest, achieving the shortest computation time, followed by the **Transformer**. All computation times were obtained with NVIDIA Tesla K80 GPU on Google's Compute Engine.

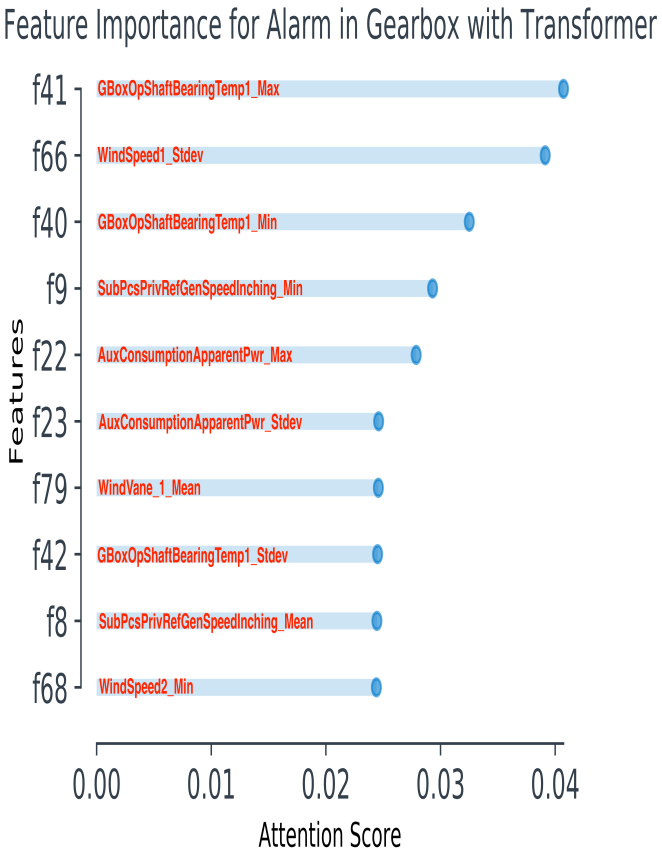

Fig. 7. Feature importance plot for an anomaly alarm in gearbox obtained with **Transformer** model

2) Error and output analysis for Stage 1: Table III shows example messages generated with each of our models alongside human references. For **Seq2Seq** we can see that while reasonable outputs are generated for the first two situations, the model recognises an error in the last situation but is not able to generate a coherent message. The **Seq2Seq (Att)** refers to the wrong sub-component in the first example, confusing the oil tank with the yaw brake. Similarly the second example confuses the pitch system with the wind direction transducer. The final message is acceptable but highlights an error as "critical" rather than "fatal". Finally, our **transformer** generates the most correct messages but still misses out on

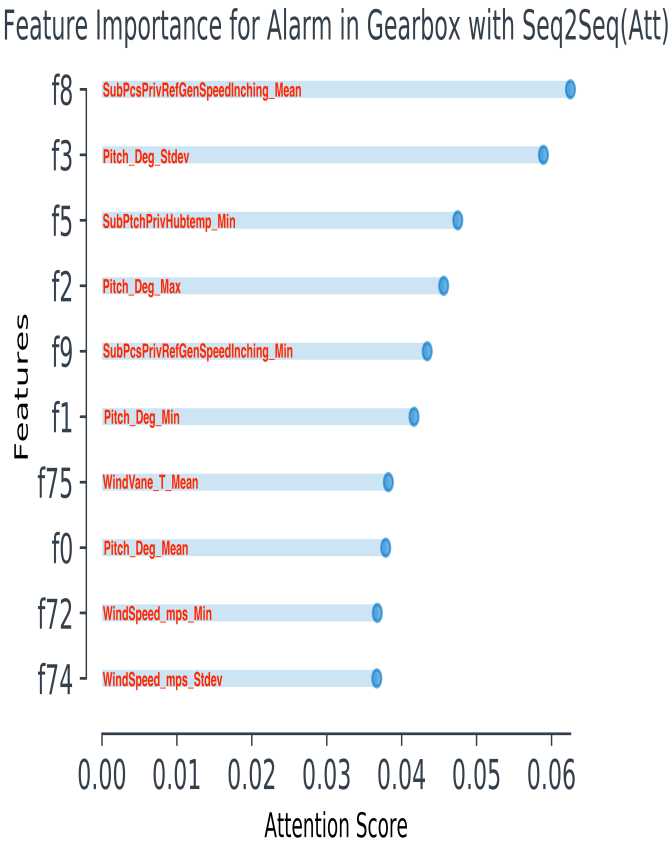

Fig. 8. Feature importance plot for an anomaly alarm in gearbox obtained with **Seq2Seq (Att)** model

nuances from the human references such as the exact tank that is being shut down as well as the error codes.

To inspect further what each of the models is doing during prediction making, Figures 7 and 8 illustrate the attention weights of the top 10 features for the example of a gearbox alarm, e.g. *demoted gearbox oil tank level shutdown*. According to the **Transformer** as in Figure 7, *GBoxOpShaftBearingTemp1_Max* and *GBoxOpShaftBearingTemp1_Min* are amongst the most highly-ranked features, which can likely be attributed to overheating of the high speed gearbox shaft bearings and the gearbox housing, thus resulting in an alarm for shutting down the oil tank due to an increasing gearbox temperature. Other features such as *SubPcsPrivRefGenSpeedInching_Min* and *AuxConsumptionApparentPwr_Max* are highly relevant based on existing literature [38] as rotational speed, efficiency and rotor speed are generally attributed as the key

Reference messages	1. (DEMOTED) Gearbox oil tank 2 level shutdown. **Alarm Code: 905** 2. Wind direction transducer error 1&3. **Alarm Code: 912** 3. Sub pitch priv fatal error has occurred more than 3 times in 86,400 Seconds. **Alarm Code: 902**
Seq2Seq	1. Demoted oil tank shutdown. **Alarm Code: 905** 2. Wind direction transducer error . **Alarm Code: 912** 3. Error occurred more than fatal in $NUM seconds.**Alarm Code: 919**
Seq2Seq (Att)	1. Demoted gearbox oil tank under pressure full brake. **Alarm Code: 901** 2. Pitch system fatal error. **Alarm Code: 919** 3. Sub pitch priv critical error has occurred more than $NUM times in $NUM seconds. **Alarm Code: 925**
Transformer	1. Demoted gearbox oil tank level shutdown. **Alarm Code: 905** 2. Wind direction transducer error .**Alarm Code: 912** 3. Sub pitch priv fatal error has occurred more than $NUM times in $NUM seconds. **Alarm Code: 902**

TABLE III
GENERATED MESSAGES FOR EACH MODEL IN STAGE 1, WITH REMARKS ABOUT THEIR VIABILITY.

causes which indirectly affect the turbine's control system, thereby the gearbox.

In contrast, feature scores for **Seq2Seq (Att)** in Figure 8 give a fair sense of features leading to the gearbox alarm. However, clearly, the relevance of the features to the alarm context is not as effective as in the **Transformer**. Specifically, *SubPcsPrivRefGenSpeedInching_Mean* signifies a high speed generator inching problem, an indirect consequence leading to contact with the generator. However, the key features from parameters pertaining to the gearbox's operational status are missed. Also, some higher-ranked features like *Pitch_Deg_Stdev* are out of context for the predicted alarm, as a disorientation of the turbine's pitch (turning angle of blades) generally has little to do with the gearbox's operation.

B. Stage 2: Maintenance strategy selection

To evaluate the performance of our Stage 2 model, which predicts the most effective maintenance actions for a fault occurring in the turbine, we held out 20% (211 samples) of our over-sampled Stage 2 data set (with 1,055 samples) for testing purposes. We compare our Stage 2 transformer against the Seq2Seq(Att) model. For any given sequence of alarm type and relevant features, we evaluate our Stage 2 model based on the number of correctly chosen maintenance actions for each held-out example.

Stage 2 Model	Percentage of maintenance actions correctly predicted
Seq2Seq (Att)	57.13%
Transformer	**75.35%**

TABLE IV
STAGE 2 RESULTS IN TERMS OF PERCENTAGE ACCURACY FOR PREDICTED MAINTENANCE ACTIONS. THE BEST PERFORMING MODEL IS SHOWN IN BOLD-FACE.

Our final model output consists of a collection of multiple natural language sub-phrases which are selected during Stage 2. Results for Stage 2 are shown in Table IV. As can be seen, our transformer model achieves an accuracy of up to 75.35%, outperforming the Seq2Seq(Att) model by up to 18.22%.

Alarm	1. Yaw error max start yaw error. 2. Blade too slow to respond. 3. Demoted gearbox filter manifold pressure shutdown. 4. Wind speed above max start.
Important Features	1. ['NacelleOrientation_Deg_Stdev', 'WindSpeed1_Max', 'WindSpeed1_Min',,'NacInsidetemp_Mean', 'NacInsidetemp_Min', 'WindSpeed1_Stdev',,'NacInsidetemp_Max', 'WindSpeed2_Max', 'WindSpeed2_Min',,'MainBearingtemp1_Stdev'] 2. ['MainBearingtemp1_Stdev', 'WindVane_T_Max', 'Pitch_Deg_Mean',,'AuxConsumptionApparentPwr_Mean', 'WindVane_T_Min',,'RotorSpeed_rpm_Max', 'RotorSpeed_rpm_Min', 'Pitch_Deg_Min',,'GearBoxTemperature_DegC_Max', 'AuxConsumptionApparentPwr_Min'] 3. ['AuxConsumptionApparentPwr_Min', 'AuxConsumptionApparentPwr_Max',,'SubIprPrivPwrFactor_Min', 'SubIprPrivPwrFactor_Mean',,'MainBearingtemp1_Max', 'MainBearingtemp1_Min',,'ReactivePower_kVAr_Stdev', 'SubIprPrivPwrFactor_Max', 'Pitch_Deg_Mean',,'Power_kW_Max'] 4. ['GearBoxTemperature_DegC_Stdev', 'WindSpeed1_Min',,'GenHeWaterInlettemp_Stdev', 'GearBoxTemperature_DegC_Max',,'NacelleOrientation_Deg_Mean', 'GearBoxTemperature_DegC_Min',,'WindSpeed_mps_Mean', 'NacInsidetemp_Max', 'Pitch_Deg_Stdev',,'WindVane_T_Stdev']
Encoded Sequence	1. [908, 58, 65, 64, 59, 60, 66, 61, 69, 68, 101] 2. [904, 101, 77, 0, 20, 76, 96, 95, 1, 53, 21] 3. [915, 21, 22, 25, 24, 100, 99, 19, 26, 0, 14] 4. [926, 54, 64, 50, 53, 55, 52, 71, 61, 3, 78]
Predicted Maintenance Action	1. a yaw misalignment is affecting the turbine operation .high and variable wind speed is affecting the stabilisation properties of the yaw brake .the wind vane measurements signal a high deviation in absolute wind direction .a forced shutdown is recommended until the absolute wind direction returns to normal . 2. blades are too slow to respond and not capturing all the wind they could .high rotational speed of the rotor is affecting spinning tension .please check the upper shaft of the rotor sweep disk for high temperature .it is suggested to use lubricants and coolants for avoiding complete failure of the rotor and yaw mechanism . 3. the turbine power performance is derated .there is wtgs deterioration occuring at present .the wind speed is very high and not in normal range .the generator oil sump temperature is not in ambient range .generator oil replacement and check is recommended . 4. the wind speed exceeds cut out speed and there is danger of damage to turbine .the wind speed is very high and not in normal range .forced shutdown is recommended until wind speed returns to normal range .the generator oil sump temperature is not in ambient range .generator oil replacement and check is recommended .

TABLE V
EXAMPLES OF PREDICTED MAINTENANCE ACTIONS USING THE DUAL-TRANSFORMER MODEL

C. Error and output analysis for Stage 2

Table V provides some examples of predicted maintenance actions selected by our second stage transformer under different scenarios. As can be observed, all output messages are grammatically coherent and fluent, given that instead of learning to generate the long sub-phrases, our second

stage transformer learns to simply select the appropriate sub-phrases from the corpus, making the learning process both simpler and more efficient. However, in some cases there were false representations of actions which were not essential to overcome the fault. Also, incorrect ordering of actions based on the fault affects relevance.

- In the first example, the turbine suffers from a fault due to yaw misalignment. Based on the relevant features causing the fault identified by our Stage 1 transformer, our model is able to effectively predict that the fault is likely being caused due to extreme variations in wind speed, which directly affects the operational consistency of the yaw brake. Finally, a forced shutdown is recommended by the model.
- In the second case, the fault occurs due to a slow response of the turbine blades. Given the features identified by the Stage 1 transformer, our model accurately produces a correct maintenance action suggestion for using lubricants in order to stabilise the bearings. Also, the relevance of the action based on the rotor speed and shaft temperature is reasonable, as the rotor hub holds the turbine blades to ensure aerodynamic efficiency, and any inconsistency in the blade movements can be attributed to difficulty in lifting of blades to make the rotor spin.
- In the third case, the model completely misses the identification of both, the relevant features as well as the maintenance action. It incorrectly predicts maintenance strategy based on derated turbine power performance, although the alarm occurred in the gearbox and is not relevant in this context.
- The final example shows a fault occurring due to wind speed being above the maximum cut-off speed to sustain turbine operation. The model is able to correctly recommend forced shutdown of the turbine until the wind speed become normal, however, in addition, an irrelevant sub-phrase for generator oil sump temperature is output. We believe this happens due to close association between the features indices of wind speed and generator oil sump temperature in our data set, and the stage 2 transformer misses the wind speed template marginally, and picks up the template nearest to it based on positioning in the predicted sequence.

Thereby, it is clear that our dual-transformer model performs very effectively in recommending the maintenance activities to be performed by the engineers and technicians, but occasionally lacks in identifying the most-relevant actions for a given sequence of SCADA features and an occurring alarm. This, despite not being sufficient for real-world use is promising considering that wind turbines are complex systems, and automated decision making still is in its infancy, and we hope to improve this in our future work.

The content-selection stage of the model thereby needs to eliminate some non-relevant sub-phrases from the output, and only output the relevant phrases to be more effective, and we believe that this can be achieved through expansion of our knowledge base beyond hand-written phrases, to include maintenance manuals and other unstructured documents.

VIII. Conclusion

This paper shows that data-to-text generation is feasible and promising for operations and maintenance in the wind industry and can assist engineers and technicians in understanding the context of an impending fault to potentially prevent it. In experiments comparing a transformer against sequence-to-sequence models with and without attention, we find that the transformer outperforms the other models in terms of performance and speed, making it a relevant model for real-time industrial application. Most importantly, it generates attention scores that are significantly more aligned with expert judgement in the domain. Our second stage transformer is able to predict the effective maintenance actions by learning from the likely causes of faults identified in the first stage. We include a brief video explaining our approach. [3] Even though our model learns mostly reasonable feature representations, there are a variety of meaningful symbols and numbers which are lost in generating alarm messages. To address this problem, we plan to explore copy actions or pointer networks in future. Further, we plan to use operational manuals for learning maintenance actions, and extend our work towards unsupervised time series summarisation for tasks wherein labelled alarm messages are not originally available.

Acknowledgments

We would like to acknowledge the Offshore Renewable Energy Catapult (OREC) for providing us access to the Levenmouth Demonstration Turbine (LDT) operational data through Platform for Operational Data (POD). We also acknowledge VIPER, the high-performance computing facility at the University of Hull and its support team, and the Aura Innovation Centre.

References

[1] C. J. Crabtree, D. Zappala, and S. I. Hogg, "Wind energy: UK experiences and offshore operational challenges," *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 229, pp. 727–746, 2015.

[2] J. Carroll, A. McDonald, I. Dinwoodie, D. McMillan, M. Revie, and I. Lazakis, "Availability, operation and maintenance costs of offshore wind turbines with different drive train configurations," *Wind Energy*, vol. 20, pp. 361–378, 2016.

[3] Y. Si, L. Qian, B. Mao, and D. Zhang, "A data-driven approach for fault detection of offshore wind turbines using random forests," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Beijing, China, October 2017, pp. 3149–3154.

[4] A. Zaher, S. McArthur, and D. Infield, "Online wind turbine fault detection through automated scada data analysis," *Wind Energy*, vol. 12, pp. 574–593, 2009.

[5] J. Chatterjee and N. Dethlefs, "Natural language generation for operations and maintenance in wind turbines," in *NeurIPS 2019 Workshop on Tackling Climate Change with Machine Learning*, Vancouver, Canada, December 2019.

[6] J. Chatterjee and N. Dethlefs, "Deep learning with knowledge transfer for explainable anomaly prediction in wind turbines," *Wind Energy*, pp. 1–18, April 2020.

[3] Brief explanation of Dual-Transformer model for predicting alarm messages and maintenance actions: https://youtu.be/HSUFzBr_mVQ

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[8] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://www.aclweb.org/anthology/D15-1166

[9] S. G. Sripada, E. Reiter, I. Davy, and K. Nilssen, "Lessons from deploying nlg technology for marine weather forecast text generation," in *Proceedings of the 16th European Conference on Artificial Intelligence*, ser. ECAI'04, 2004, pp. 760–764.

[10] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the Talk: Connecting Language Knowledge, and Action in Route Instructions," in *Proc. of National Conference on Artificial Intelligence (AAAI)*, Boston, Massachusetts, 2006.

[11] D. Chen, J. Kim, and R. Mooney, "Training a Multilingual Sportscaster: Using Perceptual Context to Learn Language," *Journal of Artificial Intelligence Research*, vol. 37, pp. 397–435, 2010.

[12] P. Vougiouklis, H. Elsahar, L.-A. Kaffee, C. Gravier, F. Laforest, J. Hare, and Elena, "Neural wikipedian: Generating textual summaries from knowledge base triples," *Journal of Web Semantics*, vol. in press, 2018.

[13] F. Mairesse, F. Jurčíček, S. Keizer, B. Thomson, K. Yu, and S. Young, "Phrase-Based Statistical Language Generation Using Graphical Models and Active Learning," in *Proc. of the 48th Annual Meeting of the Association of Computational Linguistics (ACL)*, Uppsala, Sweden, 2010.

[14] H. Hastie, H. Cuayahuitl, N. Dethlefs, S. Keizer, and X. Liu, "Evaluation of NLG in an end-to-end Spoken dialogue system-is it worth it?" in *Proc. of 7th International Workshop on Spoken Dialogue Systems*, Saariselka, Finland, 2016.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 3104–3112. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969173

[16] T.-H. Wen, M. Gašić, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[17] F. Nie, J. Wang, J.-G. Yao, R. Pan, and C.-Y. Lin, "Operations guided neural networks for high fidelity data-to-text generation," *CoRR*, vol. abs/1809.02735, 2018. [Online]. Available: http://arxiv.org/abs/1809.02735

[18] S. Gehrmann, F. Dai, H. Elder, and A. Rush, "End-to-end content and plan selection for data-to-text generation," in *Proceedings of the 11th International Conference on Natural Language Generation*, Tilburg University, The Netherlands, 2018, pp. 46–56.

[19] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *arXiv:1609.07843*. CoRR, 2016.

[20] O. Dusek and F. Jurcicek, "Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings," in *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, 2016.

[21] N. Dethlefs, "Domain Transfer for Deep Natural Language Generation from Abstract Meaning Representations," *IEEE Computational Intelligence Magazine: Special Issue on Natural Language Generation with Computational Intelligence*, 2017.

[22] M. Freitag and S. Roy, "Unsupervised natural language generation with denoising autoencoders," *CoRR*, vol. abs/1804.07899, 2018.

[23] S. Semeniuta, A. Severyn, and E. Barth, "A hybrid convolutional variational autoencoder for text generation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 627–637. [Online]. Available: http://aclweb.org/anthology/D17-1066

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional §sformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[25] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, Utah, USA, 2018.

[26] Z. Chen, H. Eavani, Y. Liu, and W. Y. Wang, "Few-shot nlg with pre-trained language model," *CoRR*, vol. arXiv:1904.09521, 2019.

[27] T. Liu, K. Wang, L. Sha, B. Chang, and Z. Sui, "Table-to-text generation by structure-aware seq2seq learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, ser. AAAI-18, 2018, pp. 4881–488.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2018. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

[29] B. Tubay and M. R. Costa-Jussà, "Neural machine translation with the transformer and multi-source romance languages for the biomedical WMT 2018 task," in *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 667–670. [Online]. Available: https://www.aclweb.org/anthology/W18-6449

[30] A. Thiruvengadam, "Transformer architecture: Attention is all you need," Mar 2019. [Online]. Available: https://medium.com/@adityathiruvengadam/transformer-architecture-attention-is-all-you-need-aeccd9f50d09

[31] J. Vig, "A multiscale visualization of attention in the transformer model," *arXiv preprint arXiv:1906.05714*, 2019. [Online]. Available: https://arxiv.org/abs/1906.05714

[32] L. Devroye, "Random sampling," *Non-Uniform Random Variate Generation*, p. 611â641, 1986.

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321â357, 2002.

[34] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[35] P. Kassebaum, "circulargraph," (https://www.github.com/paul-kassebaum-mathworks/circularGraph), 2020.

[36] M. Abadi and et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[37] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010, pp. 1045–1048.

[38] Y. Feng, Y. Qiu, C. Crabtree, H. Long, and P. Tavner, "Monitoring wind turbine gearboxes," *Wind Energy*, vol. 16, 07 2013.