Journal of Petroleum Science and Engineering 205 (2021) 108939

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

journal homepage: www.elsevier.com/locate/petrol

Automatic event identification and extraction from daily drilling reports using an expert system and artificial intelligence

Lucas P. Cinelli [a,1], José F.L. de Oliveira [a,1], Vinicius M. de Pinho [a,1], Wesley L. Passos [a,1], Rafael Padilla [a,1], Patrick F. Braz [a,1], Breno Galves [a,1], Domenica P. Dalvi [a,1], Gabriela Lewenfus [a,1], Jonathas O. Ferreira [a,1], Anthony Y.Y. Ji [b,1], Felipe L. de Oliveira [b,1], Clemente J.C. Gonçalves [b,1], Sergio L. Netto [a,1], Eduardo A.B. da Silva [a,1], Marcello L.R. de Campos [a,*,1]

[a] Electrical Engineering Program, COPPE/UFRJ, POBox 68504, Rio de Janeiro, RJ, 22941-972, Brazil
[b] Petrobras/CENPES/Drilling and Completions Research, Av. Horácio Macedo 950, Ilha do Fundão, Rio de Janeiro, RJ, 21941-915, Brazil

ARTICLE INFO

Keywords:
Automatic event extraction
Daily drilling report
Drilling data log
Natural language processing
Oil well drilling

ABSTRACT

This work addresses the problem of extracting events from human-written daily drilling reports (DDRs) in an automated way. Two distinct approaches based on an expert system and artificial intelligence techniques are proposed: rule-based language processing (RBLP) and deep neural networks (DNN). The RBLP employs regular expressions that are manually constructed, during the so-called building process, in order to identify the events of interest. The novelty of the present approach is to deal with multi-label classification of DDRs using RBLP and transformers, which provide a powerful DNN architecture. The events of interest are drilling failures such as 'bump', 'drag', 'kick', 'loss of circulation', and 'stuck pipe'. Both algorithms are developed based on a training data set of 4,355 DDRs and evaluated on a test data set of 300 DDRs, all of them written in Brazilian Portuguese but can be readily adapted/replicated to any other language. Average true positive rates (*TPR*) of 97.30% for RBLP and 85.61% for transformers-DNN were obtained, with average false negative rates (*FNR*) of 2.70% and 14.39%, respectively. The corresponding false positive rates (*FPR*) were 4.90% and 13.52%. Transformers-DNN has superior performance if the underrepresented classes are disregarded. In this case, the average *TPR* was 96.79% for RBLP and 97.32% for transformers-DNN, with an average *FNR* of 3.21% and 2.68%, respectively. The corresponding *FPR* changed to 2.37% and 1.81%. The test results indicate that the two proposed approaches can lead to very significant improvements in the efficiency of the otherwise manual annotation processes, which are typically error prone and very time consuming.

1. Introduction

Current oil-drilling operations generate a large volume of data for safety and efficiency purposes (Antoniak et al., 2016; Arnaout et al., 2014; Castiñeira et al., 2018; Kowalchuk, 2019; Noshi and Schubert, 2018; Sousa et al., 2018). Continuous drilling monitoring minimizes process interruptions and equipment damage, thus reducing non-productive time and associated losses accordingly (Sidahmed et al., 2015), which contributes to the clean development mechanism (CDM) reported in Alizadeh et al. (2014), and to more efficient resource usage as analyzed in Alizadeh et al. (2020b) and Williams et al. (2020), increasing the total factor productivity (*TFP*) (Soltanisehat et al., 2019). Thus, modern monitoring techniques also provide future competitiveness and sustainability for oil companies (Alizadeh and Soltanisehat, 2020), being key even for biomass-derived fuels (Alizadeh et al., 2020c). The collected data serve other relevant tasks in the petroleum industry, such as lithofacies modeling (Tewari and Dwivedi, 2019).

Intelligent systems for automatic fast failure detection and classification are essential components of modern monitoring schemes (Alizadeh et al., 2019). Such systems learn from experience (a list of previous failures and related sensor data) and attempt to detect anomalous behaviors as early as possible in order to mitigate their consequences, as depicted in Sidahmed et al. (2015). In general, deep technologies are increasingly being deployed in real-world applications and will be the driving force behind progress for the next couple of decades (Soltanisehat et al., 2020).

Historical data related to oil drilling operations can usually be found in drilling data logs (DDLs) and daily drilling reports (DDRs). DDLs

* Corresponding author.
E-mail address: campos@smt.ufrj.br (M.L.R. de Campos).
[1] All authors share equal responsibility in the conduction of the experiments and in the writing of the manuscript.

https://doi.org/10.1016/j.petrol.2021.108939
Received 18 January 2021; Received in revised form 7 May 2021; Accepted 10 May 2021
Available online 20 May 2021

contain drilling measurements (depth, pressure, and torque level, for example), whereas DDRs detail noticeable events (such as drill bit replacement, bump, and drag) on a daily basis. By combining these two data sources properly, one can get good patterns of all past events of interest, a fundamental step for training an accurate failure detector and classifier (Bello et al., 2016; Castiñeira et al., 2018). Besides, the collected data may steer modern data-driven approaches to engineering design that employs surrogate models to replace computationally costly simulations (Alizadeh et al., 2020a; Jia et al., 2020) with more efficient components and processes.

DDRs, however, are often written in an unstructured format, making the retrieval of information a challenging and time-consuming task (Kowalchuk, 2019; Ma et al., 2018). The need for information extraction (IE) (Small and Medsker, 2014) has driven the development of many algorithms, and previous works in the literature have attempted to address this issue by means of neural networks (Castiñeira et al., 2018; Hoffimann et al., 2018; Kowalchuk, 2019; Ribeiro et al., 2020), or random-forest algorithms (Sousa et al., 2018), obtaining reasonable event-identification performances.

The authors in Castiñeira et al. (2018), for instance, employ natural language processing solely to extract information such as hole depth and casing size for subsequent sensor measurement analysis of productive/non-productive time. In Hoffimann et al. (2018), the authors study the usage of a small convolutional neural network and long short-term memory (LSTM) network to classify the DDRs as describing an event, a symptom, or an action. Taking neighboring sentences into account for event classifications in DDRs yields good results since well-drilling operations are ordered in time (Ribeiro et al., 2020). However, the authors in Ribeiro et al. (2020) propose a complex well-specific model involving word embedding, recurrent neural networks, conditional random fields, and evolutionary algorithms, all at once. The work in Ma et al. (2018) uses 1D convolutional layers coupled with bi-directional LSTM for multi-class classification for both drilling phase and event type prediction. However, it is only able to identify a single failure during the non-productive time. In Zhang et al. (2020), the authors evaluate the performance of several machine learning methods on abnormal event detection in DDRs and claim to achieve best results when pairing convolutional neural network with techniques to overcome the inherent imbalanced data set issue.

Based on the critical evaluation of the literature, a significant gap in IE from DDRs was identified: existing models address only a single failure cause of non-productive time and/or tend to be very complex. To bridge this gap, we posed the following research question: what kind of data-driven multi-label model can perform automatic DDR event identification and extraction in a reasonably simple way? To answer this question, we investigate two distinct techniques for the automatic processing of DDRs: rule-based language processing (RBLP) and deep neural networks (DNN) (Goodfellow et al., 2016). Focus is given on the identification of five event types (drill bit 'bump', 'drag', 'kick', 'loss of circulation', and 'stuck pipe'). Events other than those aforementioned, such as problems in pumps, sudden increase or decrease in penetration, communication failure with logging while drilling (LWD) tools, are encompassed as 'other'. The results are validated over a comprehensive DDR database, achieving superior performance levels than competing approaches. To the best of our knowledge, this is the first work to deal with multi-label classification of DDRs. As said, while previous works accept only a single failure mode as the cause of non-productive time, ours embraces that multiple failure events can happen simultaneously. Another notable feature of our work is the use of neural network transformers (Vaswani et al., 2017), which are better suited to handle long-range dependencies. We propose for them a fast and straightforward training procedure with few hyperparameters. The present work compares the strengths of DNN and RBLP methods in terms of overall performance in event extraction and identification over an extensive database of DDRs.

This work is organized as follows: Section 2 presents some background on DDR content, preprocessing, and event extraction; Sections 3 and 4 describe, respectively, the proposed solutions for the automatic analysis of DDRs based on RBLP and DNN techniques; in Section 5, test results obtained with both techniques over a given DDR data set are presented and compared; and Section 6 concludes the paper emphasizing its main contributions.

2. DDR preprocessing

A typical excerpt of a DDR is given in Listing 1. It describes some drilling maneuvers in unstructured language that dealt with dragging conditions in this specific case. This listing will be used throughout the text for illustrating the steps taken during DDR processing according to our proposed methodology.

In order to facilitate its automatic analysis, the original DDR text must be preprocessed through three sequential operations: misspelling correction, lowercase conversion, and stop-word removal. It is important to note that, although the language of the database used to develop this work is the Brazilian Portuguese, the listing examples in this paper are in English, in order to show that the proposed techniques can be readily adapted to different languages.

The first procedure replaces all misspelled words with their correct versions, thus avoiding that an event such as 'drag', for instance, is missed because it was typed as 'draag' or 'darg'. Note that, in this process, some care must be taken. For example, 'darg'[2] is an existing word, but completely out of the DDR context, and therefore must be substituted by 'drag'.

In the next stage, the corrected DDR texts are converted to lowercase and have some stop words removed. Stop words are simple words like 'the', 'a', 'an', 'of', 'by', etc. that do not significantly change text meaning. When doing this operation, care must be taken for not removing words like 'with' and 'without' that give opposite meanings to the expressions they precede (as, for instance, in 'with drag' and 'without drag').

The resulting preprocessed DDRs, as exemplified in Listing 2, serve as the input to the RBLP- and DNN-based event analyzers described in Sections 3 and 4, respectively.

3. A rule-based DDR analyzer

When an event recognition or extraction is based on a set of rules, this is called a rule-processing algorithm (Thonhauser, 2004). The general form of a rule is an if-then-else statement: if *condition*, then *do-something*; else *do-something-else*. A natural language processing algorithm that employs regular expressions in a rule-based procedure is proposed in this section to automatically identify drilling events within a large DDR database.

The design of the proposed rule-based algorithm was such that it performs, in general, three (primary, secondary, and tertiary) sequential searches for regular expressions in order to decide whether a DDR describes or not a given event of interest. The primary group searches for basic keywords or expressions for each event type, which, if found, activates the corresponding event flag. If any event flag is activated after the primary search, a secondary search looks for expressions that cancel the ones in the first group (e.g., with keywords such as 'without'). If one of these is found, the event flag is turned off, and a tertiary search is performed. This tertiary search looks for expressions that contradict the secondary search (such as 'except'). If the tertiary search is successful, the event flag is activated back again. The entire search procedure is summarized in Algorithm 1. Note that the matching state of each group of expressions has to be stored. This is so because some events need a second round of primary, secondary, and tertiary searches to confirm or not the decision made by the first group. In this

Listing 1
Example of original DDR describing two drilling 'drag' events.

```
==================================================
Original DDR Text
==================================================
Open Well Pulling Drill String out of the Hole

Pulling drill string out of the hole DP 5 1/2" TT550 with 28" BHA #2 from 2411-2143 m in an open well without >>
      >> drag, except at:
- 2323-2310 m with drag of 30-50 klbf.
- 2284-2252 m with drag of 30-50 klbf.

Note:
- Total CADIT 9.9 ppg used for pulling drill string out of the hole, across the open well, equal to 1100 bbl.
```

Listing 2
Example of preprocessed DDR after misspelling correction, lowercase conversion, and stop-word removal.

```
==================================================
Preprocessed Text
==================================================
open well pulling drill string out hole

pulling drill string out hole dp 5 1/2" tt550 with bha #2 28" 2411-2143 m in open well, WITHOUT DRAG, EXCEPT:
- 2323-2310 m with drag 30-50 klbf.
- 2284-2252 m with drag 30-50 klbf.

note:
- total cadit 9.9 ppg used pulling drill string out hole, across open well, equal 1100 bbl.
```

scenario, the first group is referred to as the master and the second one as the slave.

In order to better understand how this whole process works, consider the example of identifying an event in the DDR shown in Listing 2. In this case, the primary search at the master level decides in favor of a 'drag' event, as it finds the word *drag* when using the regular expression

$$\texttt{\textbackslash b drag(s|ging|ged)?\textbackslash b}.$$

The secondary search rejects the event by finding the expression *pulling drill string out (...) without drag*, as a result of the search

$$\texttt{\textbackslash b(pulling drill string out).*(without drag)\textbackslash b}.$$

However, the tertiary group takes the final decision in favor of the event by finding the text *without drag, except* as a result of matching the regular expression

$$\texttt{\textbackslash b(without drag).*(except)\textbackslash b}.$$

This final decision seems the correct one as the DDR indicates that the drilling column is being pulled out without drag, except at some specific depths which characterize the event occurrence.

For some event types, however, such as 'drag', it may be necessary to subsequently perform the slave search round, as summarized in Algorithm 2. The slave step operates based on the result returned by the master step, and can return an opposite evaluation of event occurrence if the configured test condition is satisfied.

For events that require the slave round, the variables *ValueP*, *ValueS*, and *ValueT* determine, respectively, when and how the primary, secondary and tertiary conditions should be evaluated. The variable *ValueE* determines whether the slave round will be activated or not, depending on the value set in *bHasEvent* by the master round. If it is necessary to activate the slave round when *bHasEvent* is set to *true* by the master round, then *ValueE* is set to *true*. Likewise, if it is necessary to activate the slave round when *bHasEvent* is set to *false* by the master round, then *ValueE* is set to *false*.

If the slave step is executed, the variables *ValueP*, *ValueS*, and *ValueT* determine how the results of matching primary, secondary and tertiary regular expressions should be evaluated by the SLAVECHECKRULES function shown in Algorithm 2. If SLAVECHECKRULES returns *true*, then *bHasEvent* is set to its opposite value.

The slave action is exemplified in Listing 3. The master decided in favor of the event 'drag' because it has found the text *overpull* after matching the regular expression

$$\texttt{\textbackslash b(overpulls?)\textbackslash b}.$$

The slave step will be executed when master cycle returns *true* for the 'drag' event because *ValueE* is preset to *true* in this case. Only primary and secondary rules are available for the slave 'drag' step. Then, $ValueP = true$ and $ValueS = true$ in order to evaluate the condition

if (*primary-slave-match* is *true* **and** *secondary-slave-match* is *true*)
$$bHasEvent = false.$$

Since there are no tertiary slave rules, *bHasTerMatch* is always set to *false*. Thus, setting $ValueT = false$ has the effect of ignoring this rule. Back to the example, the slave round correctly detects that the *overpull* is not related to a 'drag' event but to a 'stuck' event. An overpull force is employed here to free the drill string. The 'stuck' event is correctly detected by the corresponding stuck-rules as shown in Listing 3.

In our training database of 4355 DDRs, only 14 (0.32%) of these had the master event status changed by the slave round.

[2] *Australian*: a fixed or definite amount of work; a work quota.

Listing 3
Example of slave algorithm in action.

```
==================================================
Original DDR Text
==================================================
Fishing / Unstuck

Connecting two pipes to allow moving Top Drive down and work on drill string.
Working on drill string with OP=50 klbf.

==================================================
Preprocessed Text
==================================================
fishing / unstuck

connecting two pipes allow moving top drive down and work drill string.
working drill string with overpull=50 klbf.

--------------------------------------------------
(X) Drag

Master Rules

Primary
Pat: \b(overpulls?)\b
Cap: overpull
--------------------------------------------------
( ) Drag

Slave Rules
The master rules accepted the event but the slave ones refused it

Primary
Pat: \b(fishing).*(/).*(unstuck)\b
Cap: fishing / unstuck

Secondary
Pat: \b(working drill string with).*(overpull)\b
Cap: working drill string with overpull
--------------------------------------------------
(X) Stuck

Master Rules

Primary
Pat: \b(fishing).*(/).*(unstuck)\b
Cap: fishing / unstuck

==================================================
Preprocessed Text -- Event Highlighted
==================================================
FISHING / UNSTUCK

connecting two pipes allow moving top drive down and work drill string.
working drill string with overpull=50 klbf.
```

3.1. Obtaining the regular expressions

Finding out the the master and slave search groups' regular expression is essentially a manual procedure performed in a developing stage. From expressions that basically capture single words that compose the set of the events of interest, such as

```
\bdrag(s|ging|ged)?\b,
```

the automatic event extraction algorithm is executed, and the results are compared with those of the manual event annotation made previously. Then, counts are generated with the events that

- **agreed with the manual annotation**, the true positives – *TP*;
- **disagreed with the manual annotation**, should have been marked as a given event, but were not, the false negatives – *FN*;
- **were added by the automatic identification and extraction**, should not have been marked as a given event, but they were, the false positives – *FP*.

During the process of obtaining the event counts, lists are also generated with the DDR identifiers for all *FN* and *FP* results. With the

Algorithm 1
Master algorithm for RBLP-based automatic event annotation.

```
# Variables that indicate the match in each group.
bool bHasPriMatch;
bool bHasSecMatch;
bool bHasTerMatch;

# Flag to store event decision.
bool bHasEvent = false;

# Daily Drilling Report text to analyze.
String X_B = DDR_Text;

# Function to reset match flags.
function RESET( )
    bHasPriMatch = false;
    bHasSecMatch = false;
    bHasTerMatch = false;
end function

# Function to match the regular expressions.
function MATCH( StringList P, StringList S, StringList T,
                String X_B )
    bHasPriMatch = HASMATCH( P, X_B );
    bHasSecMatch = HASMATCH( S, X_B );
    bHasTerMatch = HASMATCH( T, X_B );
end function

# Master part.

# Function for checking master rules.
function MASTERCHECKRULES( )
    bool bResult = false;

    if ( true == bHasPriMatch ) then
        bResult = true;
    end if

    if ( true == bHasSecMatch ) then
        bResult = false;
    end if

    if ( true == bHasTerMatch ) then
        bResult = true;
    end if

    return bResult;
end function

# Master regular expressions related to a given event.
StringList P = P_ev^1;          # Primary regular expressions.
StringList S = S_ev^1;          # Secondary regular expressions.
StringList T = T_ev^1;          # Tertiary regular expressions.

RESET( );
MATCH( P, S, T, X_B );

# Store master decision.
bHasEvent = MASTERCHECKRULES( );
```

aid of a graphical user interface, all incorrectly evaluated DDRs are (re)examined, and a new set of regular expressions is derived. As a result of this process, it was possible to establish regular expression sets for all events registered in a training set of 4355 manually annotated DDRs, as quantified in Table 1. These expressions were then tested on an independent set of 300 DDRs, as detailed in the experiments in Section 5.

4. A neural-network DDR analyzer

Deep learning is the common term for the class of artificial-intelligence algorithms based on neural networks with many layers. While it may seem trivial to use more layers, there are several technical and computational challenges involved.

After the preprocessing stage described in Section 2, the sentences are split into meaningful chunks, called tokens. This process, referred

Algorithm 2
Slave algorithm for RBLP-based automatic event annotation.

```
# Master part, from Algorithm 1.

# Master regular expressions related to a given event.
P = P_ev^1;
S = S_ev^1;
T = T_ev^1;

RESET( );
MATCH( P, S, T, X_B );

# Store master decision.
bHasEvent = MASTERCHECKRULES( );

# Slave Part.

# Function for checking slave rules.
function SLAVECHECKRULES( bool bPriValue,
                          bool bSecValue,
                          bool bTerValue )
    if ( bPriValue != bHasPriMatch ) then
        return false;
    end if

    if ( bSecValue != bHasSecMatch ) then
        return false;
    end if

    if ( bTerValue != bHasTerMatch ) then
        return false;
    end if

    return true;
end function

# ValueE, ValueP, ValueS and ValueT below can be set true
# or false, depending on the event being analyzed.

if ( ValueE == bHasEvent ) then
    # Slave regular expressions related to a given event.
    P = P_ev^2;                 # Primary regular expressions.
    S = S_ev^2;                 # Secondary regular expressions.
    T = T_ev^2;                 # Tertiary regular expressions.

    RESET( );
    MATCH( P, S, T, X_B );

    if ( true == SLAVECHECKRULES( ValueP, ValueS, ValueT ) ) then
        # Change master decision.
        bHasEvent = !bHasEvent;
    end if
end if
```

Table 1
Number of regular expressions used for the RBLP algorithm per event type for each search group (Pri = Primary, Sec = Secondary, Ter = Tertiary) and level(M = master, S = slave).

Event	Pri M	Sec M	Ter M	Pri S	Sec S	Ter S	Total
Bump	3	0	0	0	0	0	3
Drag	11	63	15	2	3	1	95
Kick	3	8	0	0	0	0	11
Loss	3	60	6	2	3	0	74
Stuck	17	10	14	1	5	1	48
Other	403	22	2	0	0	0	427
Total	440	163	37	5	11	2	**658**

to as tokenization (Jurafsky and Martin, 2009), can happen at character, word, or subword level. While character-level tokenization is too simplistic and requires the deep neural networks (DNNs) to have significantly more parameters to deal with the conceptual grouping of characters, word-level tokenization represents an exponentially large-dimension space. Subword tokenization is a good compromise that combines word and character level tokenization strengths while alleviating their drawbacks: it achieves a semantically meaningful space while keeping dimensionality under control.

The common bottleneck for algorithms based on DNNs is the lack of sufficient annotated data for system training. Recent works in the field of natural language processing have tackled this issue by using the bidirectional encoder representations from transformers (BERT) algorithm (Devlin et al., 2019).

In the context of NLP, a transformer is a type of DNN that relies on an encoder–decoder structure that forgoes recurrent architectures and enables significant parallelization. The encoder converts the input sequence into an intermediate representation containing information from a potentially vast context. Stacking multiple encoders allows the DNN to learn more complex and relevant relationships through training.

A key component of the BERT algorithm is the concept of self-attention that is used to map one sequence into another (Vaswani et al., 2017). Attention is a mechanism that determines which parts of an input sequence are more relevant to a given task, and self-attention uses the input sequence itself to determine its more relevant parts. In this scheme, there are two steps in the system optimization process: pre-training and fine-tuning.

The model is adjusted to unlabeled data over two different tasks during pre-training: masked language model and next sentence prediction. In the former, the model must predict randomly selected tokens within the sequence (not the next or previous token). In the latter, the model must predict if the two sentences stitched together are actually contiguous or not. During the fine-tuning step, all model weights are refined.

When employing the BERT algorithm in DDR analysis, the pre-training for the Brazilian Portuguese language model was derived based on the large text corpus described in Wagner Filho et al. (2018), whereas the fine-tuning for drilling event identification was based on the subset of 4355 manually annotated DDRs.

The general steps employed in the DNN-based text-processing algorithm are text preprocessing as explained in Section 2, tokenization, numericalization, embedding, and classification.

The present work uses SentencePiece (Kudo and Richardson, 2018), a subword tokenizer specific for DNN-based text processing, coupled with byte pair encoding (BPE) (Sennrich et al., 2016). In BPE, the vocabulary starts with single characters, but after initially tokenizing the text, the most frequent pairs of symbols are merged and included in the vocabulary.

Numericalization consists of transforming the tokens into integers according to the dictionary built during the previous step (Jurafsky and Martin, 2009). The dictionary represents the set of all known tokens, that is, the vocabulary. A special token replaces tokens outside the vocabulary.

The embedding step maps the high-dimensional sparse one-hot encoding of integers (Vidgen et al., 2019) onto computationally efficient dense real-numbered vectors. The transformation is learned from training on a large corpus. The optimization creates a structured space where semantically similar tokens are mapped onto close vectors.

Finally, the embedding matrix, a concatenation of all embedding vectors of a sample sentence, is fed to the DNN-based classifier, which outputs a confidence value c_f within the interval $[0, 1]$ for each failure type f. If the value is above a specified threshold t_f, the input sample is predicted as positive for the corresponding failure f. If no failure is predicted, the sample is considered 'normal'.

The present work uses the same BERT base architecture defined in Devlin et al. (2019), with 12 layers, 12 attention heads, 768 hidden units at each layer, and a total of 109 million trainable parameters. We used the Portuguese pre-trained model made publicly available in Souza et al. (2019). The training was performed for 1 million steps, which amount to approximately 8 epochs over the training data set, a process that lasted 4 days on a TPUv3-8 instance. Their model builds on the multilingual BERT base weights, which was also trained for 4 days but on 16 TPU chips instead. Hence, most of the hyperparameters have already been fixed, needing only to set a few, e.g., learning rate and regularization strength. The vocabulary consists of about 30,000 cased subword units and was extracted from a set of 200,000 articles from the Brazilian Wikipedia (Souza et al., 2019).

We also fine-tuned the Portuguese base model for multi-label DDR classification by training on 4355 DDR samples for 6 epochs, which lasts around 14 min. As the loss function, we used the weighted binary cross-entropy loss function

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\frac{1}{N}\sum_{i=0}^{N-1} w_i \left[y_i \log\left(\sigma(x_i)\right) + (1 - y_i)\log\left(1 - \sigma(x_i)\right)\right], \quad (1)$$

where w_i are the class weights, x_i and y_i the predicted and target binary output for class i, respectively, and $\sigma(x_i) = 1/(1+\exp^{-x_i})$ is the sigmoid function.

Binary cross-entropy treats each output independently, so it is suitable for multi-label classification use cases. Since the training set is heavily skewed with some classes being 10 to 100 times more frequent than others, we use the inverse class frequencies as the $\mathbf{w}$ class weights.

5. Experimental results

In this section, the results obtained by the RBLP- and DNN-based algorithms are presented and compared. The advantages and disadvantages of both methods are also pointed out. All results that follow were obtained by the systems trained with 4355 manually-annotated DDRs from 55 different wells. The testing data set consisted of an additional 300 manually-annotated DDRs from a single well apart from the training set.

5.1. RBLP-based event identification

Table 2 shows the test results obtained by the RBLP-based DDR analyzer. In this table, one readily notices the good performances achieved in automatically identifying all specific failure classes (different from the class 'other'), with a few *FP* cases except in the 'other' and 'normal' classes.

The RBLP method encompasses 36 types of events in the 'other' class such as pump, top-drive, and drill bit failures. For each type, a group of regular expressions was built that could properly match it. However, in the 'other' class, there was no generic type that could simply report that, in a given DDR, there are descriptions of failures and/or problems other than those already contemplated in this set of 36 types. If a more generic regular expression is adopted by adding

```
\bfailures?\b
```

and testing it only if the specific events 'other' are not found first, better results for this class can be obtained with the *TP* rate (*TPR*) increasing to 96.64% (115 out of 119 events) and the *FN* rate (*FNR*) decreasing to 3.36% (only 4 events instead of 94), as shown in Table 3. The overall number of *FN* and *FP* has also decreased.

Note that this change was not introduced based on reading the test sets' DDRs, but conjecturing about the possible causes of worse performance for the class 'other' obtained by the RBLP method.

Another relevant information is that only one out of the 300 DDRs in the test data set had the master event status changed by the slave round, indicating, once again, the low criticality of that stage.

5.2. DNN-based event identification

The neural network outputs confidence values in the $[0, 1]$ range for each failure class. The higher the value, the more confident the model is in its prediction. It is, therefore, necessary to set, for each type of failure, probability thresholds that, if exceeded, will characterize a positive. The freedom to choose these thresholds enables the user to establish the relative importance between *FP* and *FN*, implicitly

Table 2
Performance breakdown for the RBLP event annotator in the test set.

Event	Count	*TP* (%)	*TP*	*FN* (%)	*FN*	*FP* (%)	*FP*
Bump	4	100.00	4	0.00	0	0.34	1
Drag	47	97.87	46	2.13	1	5.14	13
Kick	2	100.00	2	0.00	0	0.00	0
Loss	28	89.29	25	10.71	3	1.10	3
Stuck	25	100.00	25	0.00	0	2.91	8
Other	119	21.01	25	78.99	94	10.50	19
Normal	96	85.42	82	14.58	14	39.71	81

Table 3
Improved performance breakdown for the RBLP event annotator in the test set.

Event	Count	*TP* (%)	*TP*	*FN* (%)	*FN*	*FP* (%)	*FP*
Bump	4	100.00	4	0.00	0	0.34	1
Drag	47	97.87	46	2.13	1	5.14	13
Kick	2	100.00	2	0.00	0	0.00	0
Loss	28	89.29	25	10.71	3	1.10	3
Stuck	25	100.00	25	0.00	0	2.91	8
Other	119	**96.64**	**115**	**3.36**	**4**	19.89	36
Normal	96	85.42	82	14.58	14	**1.96**	**4**

Table 4
Performance breakdown for the DNN event annotator in the test set.

Event	Count	Threshold	*TP* (%)	*TP*	*FN* (%)	*FN*	*FP* (%)	*FP*
Bump	4	1.98×10^{-1}	100.00	4	0.00	0	0.00	0
Drag	47	2.16×10^{-1}	100.00	47	0.00	0	3.95	10
Kick	2	3.36×10^{-3}	100.00	2	0.00	0	67.79	202
Loss	28	6.40×10^{-1}	89.29	25	10.71	3	0.37	1
Stuck	25	4.63×10^{-1}	100.00	25	0.00	0	2.91	8
Other	119	9.84×10^{-1}	24.37	29	75.63	90	6.08	11
Normal	96	–	29.17	28	70.83	68	33.82	69

trading-off detection and false alarm rates. If the ultimate goal is a semi-automatic procedure in which a human expert assesses the DDR predicted with some failure, then the aim is to reduce DDR overload without compromising overall assertiveness. Thus, one should opt for high detection rates to the detriment of false alarm rates.

The adequate tool for analyzing the performance of methods with a variable threshold is the receiver operating curve (ROC). This graphical representation illustrates a binary classifier's performance in terms of its *TP* and *FP* predictions as its threshold varies. In the multi-label case, in which the same DDR can simultaneously have different labels, one ROC per label is used, as given in Fig. 1.

Table 4 exhibits the performance of the DNN-based event extractor in the test set for the same *TP* level as RBLP-based DDR analyzer discussed in the previous section. The model performs very well for most types of failure, with the exception of those of the classes 'kick', which has very few instances, and 'others', which includes several non-targeted event types. The 'normal' class achieves a low performance because it does not have a dedicated tag during training, as it simply indicates a no-failure status.

5.3. Overall RBLP and DNN comparison

An overall comparison between RBLP and DNN results is presented here. Tables 5 and 6 show the average results compiled from those obtained in the previous two subsections. The rates on the first pair of columns, RBLP and DNN, include all events of interest. The rates on the second pair exclude the 'normal' event, and the rates of the last pair exclude 'kick', 'other' and 'normal' events. This is done to take into account the notes made in Section 5.2 for the worse performance of DNN concerning these cases. As it can be seen in Table 5, if the 'normal' event is excluded, DNN has superior *TPR* and *FNR*, but worse performance in terms of *FPR*. In relation to the improved RBLP (denoted herein RBLP*), shown in Table 6, DNN can only reach superior

Table 5
Average RBLP and DNN performance. The minus sign (–) means 'excluding event'.

Rate (%)	RBLP	DNN	RBLP – Normal	DNN – Normal	RBLP – Kick, Other, Normal	DNN – Kick, Other, Normal
TP	84.80	77.55	84.69	85.61	96.79	97.32
FN	15.20	22.45	15.31	14.39	3.21	2.68
FP	8.53	16.42	3.33	13.52	2.37	1.81

Table 6
Average RBLP* (improved RBLP) and DNN performance. The minus sign (–) means 'excluding event(s)'.

Rate (%)	RBLP*	DNN	RBLP* – Normal	DNN – Normal	RBLP* – Kick, Other, Normal	DNN – Kick, Other, Normal
TP	95.60	77.55	97.30	85.61	96.79	97.32
FN	4.40	22.45	2.70	14.39	3.21	2.68
FP	4.48	16.42	4.90	13.52	2.37	1.81

performance if the problematic 'kick', 'other' and 'normal' events are not used to compute the rates, as already explained in Section 5.2.

The identification and extraction of events based on DNN offer flexibility in terms of the trade-off between detection and false alarm rates obtained via the threshold adjustment translated as different $TP \times FP$ operating points in the ROC curve. However, flexibility means extra model complexity and the accompanying requirement of more data for training the network. This meant relatively poorer performance in our system for detecting 'kicks' and non-targeted event types, treated as 'other.' On the other hand, the RBLP approach designed upon a carefully chosen master–slave iterative protocol could perform very well even for the highly unbalanced data-starving training set. Its overall good performance was achieved at the price of lack of flexibility in terms of the trade-off between detection and false alarm rates, shown as a single operating point instead of a full ROC curve.

Our results using real-life data indicate that the two approaches can be employed to put together a powerful toolbox for the post-drilling processing of DDRs, in which one can choose the desired compromise between computational complexity and operating point flexibility. Its use can greatly improve the efficiency of the otherwise time-consuming and error-prone manual activity of going through large log files, typically equivalent to finding needles in a haystack.

6. Conclusion

In this article, a significant gap in IE from daily drilling reports (DDRs) was addressed: to develop a robust multi-label data-driven model for automatic DDR event identification and extraction. To bridge this gap, two new approaches were considered: a rule-based language processing (RBLP) algorithm and a deep neural network (DNN). The RBLP employs regular expressions that are manually constructed to extract/identify the events of interest. To the authors' best knowledge, a method like this was never used to build a multi-label event identifier with such a notably good performance. The DNN system uses a pre-training stage based on a large (non-annotated) data set and fine-tunes the algorithm on the same manually annotated DDR data set as the RBLP system. Although there are works on multi-class DDR classification using DNN architectures, such as LSTMs, this is the first to use transformer, which carries several advantages, such as: avoid recursion, are highly parallelizable, better capture long-range dependencies, making them an excellent choice for text classification.

Both systems were tested and compared using a set of 300 manually annotated DDRs for five common events indicating failures. The main results obtained reveal an average *TPR* of 97.30% for RBLP and 85.61% for transformers-DNN with average *FNR* of 2.70% and 14.39%, respectively, and a corresponding *FPR* of 4.90% and 13.52%. Transformers-DNN had superior performance when the problematic

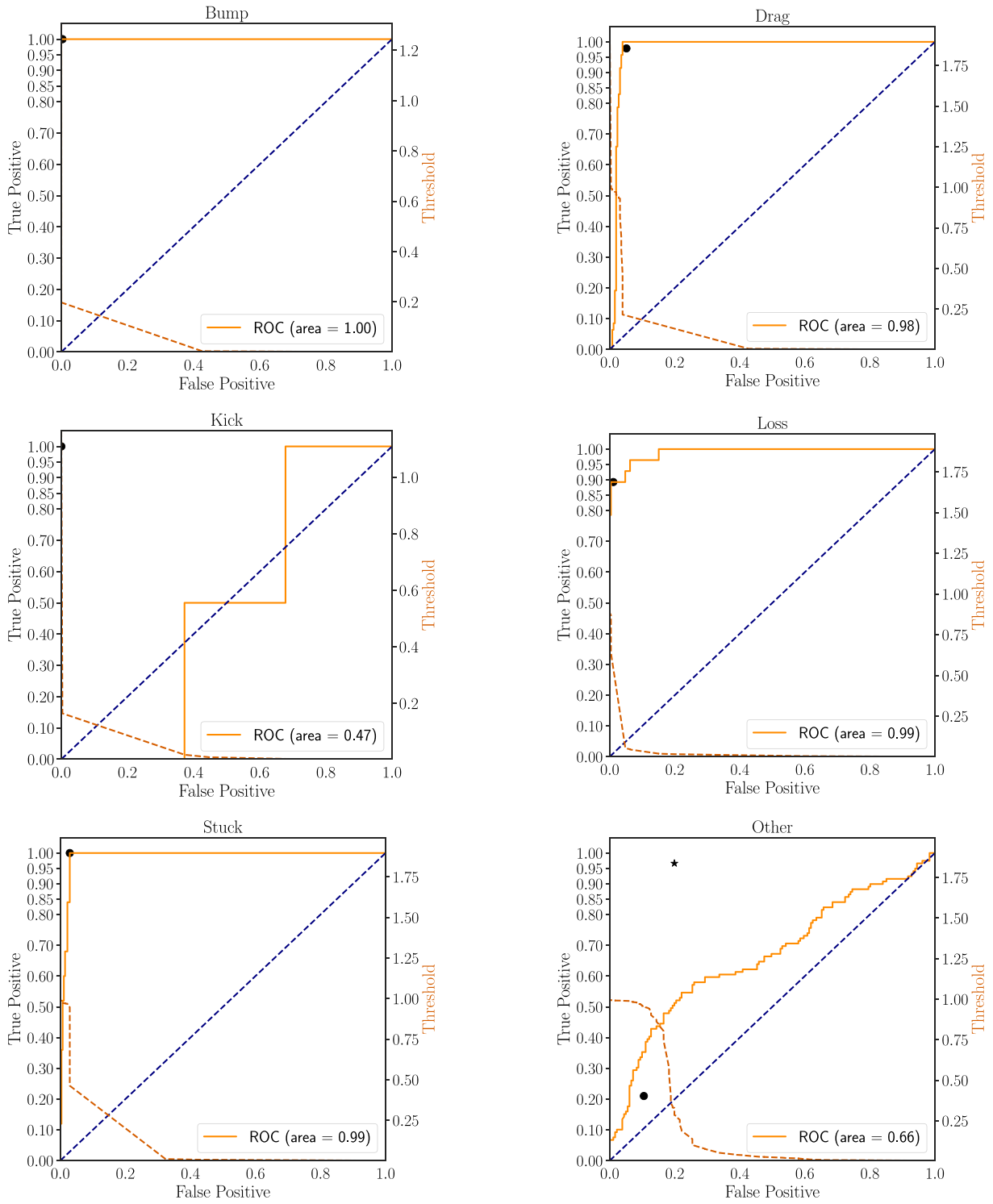

Fig. 1. ROC performances (solid orange line) for the DNN-based event annotator in the test set. The dashed orange line represents the threshold for each false positive value. Performance of the RBLP-based annotator is represented by the black dot. The black star depicts the performance of the improved RBLP for the 'other' class. Since only this class was affected by the improvement, only black dots appear on the remaining plots.

cases, with very few training samples, were excluded when computing rates. In this case, the average *TPR* was 96.79% for RBLP and 97.32% for transformers-DNN, with an average *FNR* of 3.21% and 2.68%, respectively, and a corresponding *FPR* of 2.37% and 1.81%. Altogether, such results indicate that the multiclass problem of automatic DDR analysis has been successfully addressed by both proposed approaches.

As a future direction, one may consider employing pre-trained word embedding models to leverage the domain knowledge onto the final event extractors. Another possibility is to address the class imbalance issue that hinders DNN optimization and negatively impacts classification performance, as demonstrated in Zhang et al. (2020), by employing synthetic minority over-sampling technique (SMOTE) (Chawla et al., 2002) and focal loss (Lin et al., 2017).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. It was also partly funded by CNPq, Brazil, FAPERJ, Brazil, and Petrobras, Brazil (PT-101.01.13012).

References

Alizadeh, R., Allen, J.K., Mistree, F., 2020a. Managing computational complexity using surrogate models: a critical review. Res. Eng. Des. 31, 275–298. http://dx.doi.org/10.1007/s00163-020-00336-7.

Alizadeh, R., Beiragh, R.G., Soltanisehat, L., Soltanzadeh, E., Lund, P.D., 2020b. Performance evaluation of complex electricity generation systems: A dynamic network-based data envelopment analysis approach. Energy Econ. 91, 1–15. http://dx.doi.org/10.1016/j.eneco.2020.104894.

Alizadeh, R., Jia, L., Nellippallil, A.B., Wang, G., Hao, J., Allen, J.K., Mistree, F., 2019. Ensemble of surrogates and cross-validation for rapid and accurate predictions using small data sets. Artif. Intell. Eng. Des. Anal. Manuf. 33, 484–501. http://dx.doi.org/10.1017/S089006041900026X.

Alizadeh, R., Lund, P.D., Soltanisehat, L., 2020c. Outlook on biofuels in future studies: A systematic literature review. Renew. Sustain. Energy Rev. 134, 1–14. http://dx.doi.org/10.1016/j.rser.2020.110326.

Alizadeh, R., Maknoon, R., Majidpour, M., 2014. Clean development mechanism, a bridge to mitigate the greenhouse gasses: Is it broke in iran?. In: 13th International Conference on Clean Energy – ICCE 2014, Istanbul, Turkey. pp. 399–404.

Alizadeh, R., Soltanisehat, L., 2020. Stay competitive in 2035: a scenario-based method to foresight in the design and manufacturing industry. Foresight 22, 309–330. http://dx.doi.org/10.1108/FS-06-2019-0048.

Antoniak, M., Dalgliesh, J., Verkruyse, M., Lo, J., 2016. Natural language processing techniques on oil and gas drilling data. In: SPE Intelligent Energy International Conference and Exhibition, SPE-181015-MS. Aberdeen, Scotland, United Kingdown, pp. 1–6. http://dx.doi.org/10.2118/181015-MS.

Arnaout, A., O'Leary, P., Esmael, B., Thonhauser, G., 2014. Distributed recognition system for drilling events detection and classification. Int. J. Hybrid Intell. Syst. 11, 25–39. http://dx.doi.org/10.3233/HIS-130181.

Bello, O., Teodoriu, C., Yaqoob, T., Oppelt, J., Holzmann, J., Obiwanne, A., 2016. Application of artificial intelligence techniques in drilling system design and operations: A state of the art review and future research pathways. In: SPE Nigeria Annual International Conference and Exhibition, SPE-184320-MS. Lagos, Nigeria, pp. 1–22. http://dx.doi.org/10.2118/184320-MS.

Castiñeira, D., Toronyi, R., Saleri, N., 2018. Machine learning and natural language processing for automated analysis of drilling and completion data. In: SPE Kingdown of Saudi Arabia Annual Technical Symposium and Exhibition, SPE-192280-MS. Dammam, Saudi Arabia, pp. 1–16. http://dx.doi.org/10.2118/192280-MS.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artificial Intelligence Res. 16, 321–357.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1. Minneapolis, USA, pp. 4171–4186. http://dx.doi.org/10.18653/v1/N19-1423.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. In: Adaptive Computation and Machine Learning, MIT Press, Cambridge, USA, URL: https://www.deeplearningbook.org/.

Hoffimann, J., Mao, Y., Wesley, A., Taylor, A., 2018. Sequence mining and pattern analysis in drilling reports with deep natural language processing. In: SPE Annual Technical Conference and Exhibition, SPE-191505-MS. Dallas, USA, pp. 1–7. http://dx.doi.org/10.2118/191505-MS.

Jia, L., Alizadeh, R., Hao, J., Wang, G., Allen, J.K., Mistree, F., 2020. A rule-based method for automated surrogate model selection. Adv. Eng. Inform. 45, 1–16. http://dx.doi.org/10.1016/j.aei.2020.101123.

Jurafsky, D., Martin, J.H., 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, second ed. Pearson Prentice Hall, Upper Saddle River, New Jersey, USA.

Kowalchuk, P., 2019. Implementing a drilling reporting data mining tool using natural language processing sentiment analysis techniques. In: SPE Middle East Oil and Gas Show and Conference, SPE-194961-MS. Manama, Bahrain, pp. 1–14. http://dx.doi.org/10.2118/194961-MS.

Kudo, T., Richardson, J., 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Brussels, Belgium, pp. 66–71. http://dx.doi.org/10.18653/v1/D18-2012.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: International Conference on Computer Vision, Venice, Italy. pp. 2999–3007.

Ma, Z., Vajargah, A.K., Lee, H., Darabi, H., Castineira, D., 2018. Applications of machine learning and data mining in speedwise® drilling analytics: A case study. In: Abu Dhabi International Petroleum Exhibition & Conference, SPE-193224-MS. Abu Dhabi, UAE, pp. 1–19. http://dx.doi.org/10.2118/193224-MS.

Noshi, C.I., Schubert, J.J., 2018. The role of machine learning in drilling operations; a review. In: SPE/AAPG Eastern Regional Meeting, SPE-191823-18ERM-MS. Pittsburgh, Pennsylvania, USA, pp. 1–16. http://dx.doi.org/10.2118/191823-18ERM-MS.

Ribeiro, L.C., Afonso, L.C., Colombo, D., Guilherme, I.R., Papa, J.P., 2020. Evolving neural conditional random fields for drilling report classification. J. Pet. Sci. Eng. 187. http://dx.doi.org/10.1016/j.petrol.2019.106846.

Sennrich, R., Haddow, B., Birch, A., 2016. Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany, pp. 1715–1725. http://dx.doi.org/10.18653/v1/P16-1162.

Sidahmed, M., Coley, C.J., Shirzadi, S., 2015. Augmenting operations monitoring by mining unstructured drilling reports. In: SPE Digital Energy Conference and Exhibition, SPE-173429-MS. The Woodlands, Texas, USA, pp. 1–13, doi: SPE-173429-MS.

Small, S.G., Medsker, L., 2014. Review of information extraction technologies and applications. Neural Comput. Appl. 25, 533–548. http://dx.doi.org/10.1007/s00521-013-1516-6.

Soltanisehat, L., Alizadeh, R., Hao, H., Choo, K.K.R., 2020. Technical, temporal, and spatial research challenges and opportunities in blockchain-based healthcare: A systematic literature review. IEEE Trans. Eng. Manage. 1–16. http://dx.doi.org/10.1109/TEM.2020.3013507.

Soltanisehat, L., Alizadeh, R., Mehregan, N., 2019. Research and development investment and productivity growth in firms with different levels of technology. In: Iranian Economic Review – IER, Vol. 23. Faculty of Economics, University of Tehran, Tehran, Iran, pp. 795–818.

Sousa, G.J., Pedronette, D.C.G., Baldassin, a., Privatto, P.I.M., Gaseta, M., Guilherme, I.R., Colombo, D., Afonso, L.C.S., Papa, J.P., 2018. Pattern analysis in drilling reports using optimum-path forest. In: 2018 International Joint Conference on Neural Networks – IJCNN. Rio de Janeiro, Brazil, pp. 1–8. http://dx.doi.org/10.1109/IJCNN.2018.8489232.

Souza, F., Nogueira, R., Lotufo, R., 2019. Portuguese named entity recognition using BERT-CRF. arXiv preprint arXiv:1909.10649 URL: http://arxiv.org/abs/1909.10649.

Tewari, S., Dwivedi, U.D., 2019. Ensemble-based big data analytics of lithofacies for automatic development of petroleum reservoirs. Comput. Ind. Eng. 128, 937–947. http://dx.doi.org/10.1016/j.cie.2018.08.018.

Thonhauser, G., 2004. Using real-time data for automated drilling performance analysis. In: Spring Meeting of DGMK and OEGEW 2004, Celle. pp. 170–173.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems 30, Long Beach, USA. pp. 5998–6008.

Vidgen, R., Kirshner, S., Tan, F., 2019. Business Analytics: A Management Approach. Red Globe Press, London, UK.

Wagner Filho, R., Idiart, M., Villavicencio, A., 2018. The brWaC corpus: A new open resource for brazilian portuguese. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation – LREC 2018, Miyazaki, Japan. pp. 4339–4344.

Williams, J., Alizadeh, R., Allen, J.K., Mistree, F., 2020. Using network partitioning to design a green supply chain. In: Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference – IDETC/CIE 2020, Virtual, Online. p. Volume 11B. http://dx.doi.org/10.1115/DETC2020-22644.

Zhang, H., Zeng, Y., Bao, H., Liao, L., Song, J., Huang, Z., Chen, X., Wang, Z., Xu, Y., Jin, X., 2020. Drilling and completion anomaly detection in daily reports by deep learning and natural language processing techniques. In: SPE/AAPG/SEG Unconventional Resources Technology Conference, URTEC-2020-2885-MS, Virtual. pp. 1–10. http://dx.doi.org/10.15530/urtec-2020-2885.