Algorithmically Exploiting the Knowledge Accumulated in Textual Domains for Technical Support

Daniele Pau[1], Isaia Tarquini[2], Matteo Iannitelli[3], Carmine Allegorico[4]

[1,3,4]*Baker Hughes, Firenze, 50127, Italy*
name.surname@bakerhughes.com

[2]*independent researcher, Bologna, 40137, Italy*
isaia.tarquini@gmail.com

ABSTRACT

Extracting relevant information from natural language, in the deep domain of technical engineering, remains an open challenge for Natural Language Processing (NLP).

On the other side, Original Equipment Manufacturers (OEM) produce an ever-increasing number of textual documents during the entire lifecycle of products to support the execution of many activities and services, such as maintenance, management of spare parts, remote Monitoring and Diagnostics (M&D), etc.

This paper presents a case study in which Artificial Intelligence (AI) algorithms were applied to extract knowledge from textual data of unlabeled technical cases, thus helping subject matter experts when approaching new cases.

In the first part of this work, we present the operating process and the available textual data; the focus is on the outbound communication delivered from the technical team to the site operators, which is structured in three main sections: event description, technical assessment, recommended actions.

The work proceeded in two parallel streams using two different datasets: the first concerned with the analysis of event descriptions and technical assessments, aiming to detect recurring topics; the second involved the recommended actions, with the goal to create a library of recommendations, which may enable standardization and the creation of new AI/NLP developments.

A tailored text preprocessing activity was applied to both datasets, to define relevant domain entities and stopwords, map acronyms and synonyms for context disambiguation, split sentence for recommended actions, and finally to lemmatize text.

The subsequent steps for feature extraction were performed using bag of words (TF-IDF) and word embeddings (W2V, BERT) models to transform the language domain into vectors in multidimensional numeric domain. Afterwards, data has been split into homogenous groups, testing multiple unsupervised learning algorithms, such as: LDA, k-means, spectral clustering, affinity propagation and HDBSCAN. Each combination between language model and clustering technique was evaluated using the silhouette score and visual analysis.

To validate the effectiveness, the developed NLP algorithms were deployed as web services and integrated into the digital framework used by the technical support experts to deliver the service. A dedicated web app was also developed, to show trending topics and retrieve insightful information on clusters.

Finally, an outlook on the open technical challenges and future perspective of the ongoing work is presented in the conclusions.

1. INTRODUCTION

Manufacturers of high-value technology products, such as rotating equipment serving in critical industrial applications, usually offer after-sales services to ensure availability and reliability of the assets over years (Iannitelli et al., 2018).

After-sales services can be performed directly on site, like periodic maintenance or Field Service activities, by OEM technical experts residing on site, or can be delivered remotely, e.g. through Monitoring and Diagnostic and technical product service. In both cases, OEM generates written technical documents: for instance, maintenance is managed through Work Orders, documents detailing list of activities, spare parts, consumables, costs, etc.; technical experts working at site usually produce periodic reports to summarize all encountered issues and performed actions; M&D and Product Service provide to site operators, proactively or under request, technical relations on detected anomalies accompanied by recommended actions to do. In any case, a considerable amount of technical engineering textual data is generated and stored, mostly as unstructured data, in manufacturers databases.

In Baker Hughes we have developed monitoring capabilities offered as a service (iCenter) that we apply to a broad installed fleet of rotating equipment; and the technical

insights delivered to site operators by M&D department are the focus of this work.

The work process of Baker Hughes iCenter can be described as follows: data from sensors installed on machines is acquired and stored in databases/cloud and feed a set of analytics, checking for incipient or occurred functional failures. If a real issue is verified, M&D prepares and sends to site operators a technical report including the description of the event occurred, a technical assessment, and the recommendations/dispositions for the site personnel (Iannitelli et al., 2018).

NLP applied to engineering technical domains, also known as Technical Language Processing (Brundage et al., 2020), is an interesting research topic. One of the main applications is the processing of maintenance work orders and relevant works can be found in literature (Stenström et al. 2015; Sexton et al. 2018, Sexton et al. 2019; Sharp et al. 2021).

In this work we present a real use case of natural language processing applied to engineering technical reports, with the aim of extracting valuable *know-how* and increase the quality of the service provided. The final aim of this work is to improve the already existing IT tools in order to support technical experts during troubleshooting activities or the definition of recommendations for site operators. The expected medium-term outcomes, for which this paper represents a first step, are:

- for site operators, receive more standardized technical insights, more precise and with higher quality;
- for M&D operators, supporting junior professionals, increasing productivity and the relevance of technical assessments.

In particular, new proactive tools can be developed to suggest relevant past issues or highlight similar cases previously managed, which leads to prevent human errors and give better insights. On this track, we developed a web application, presented in paragraph 6, with the aim to test the NLP outcomes and validate the M&D use cases. This web app is fully integrated in the current iCenter SW infrastructure and represents a first effort for future extensions.

Processing the natural language would also enable deeper statistical analysis and reporting. For instance, discovery of recurrent anomalies would also benefit the broader company, since feedbacks to sourcing, design or manufacturing departments can be derived after identifying the root cause of such issues.

2. EXPLORATORY DATA ANALYSIS

Once an anomaly has been detected and a technical insight sent to site operators, the textual content of the notification is stored in a database. Each record contains a number of structured information, such as the expert who made the technical analysis, creation date, issue category (e.g. combustion problems, lubrication, instrument failure) and three sections manually filled by the expert, which represent the main data source of this work: the event description, summarizing the event highlights, the technical assessment, reporting the functional failures (incipient or occurred), and the relevant troubleshooting and recommendations, a set of actionable dispositions to solve the issue or mitigate the risk. In Table **1** is reported an example of technical notification, regarding an instrument issue.

Table 1: example of technical notification

Description	An anomalous behavior was observed on cooling water tank level 212L200
Technical Assessment	Trends show that cooling water reservoir level was showing a reliable reading around 90% with oscillations correlated to ambient temperature variation. On 8/1/2019 the reading showed increases to 100%, followed by spikes to 0 and a constant reading around 80% with no influence by ambient temperature. No change observed in other cooling water parameters such as header temperature and pressure. The behavior could be related to an Issue on cooling water tank level 212L200.
Troubleshoot	- Check the acquisition loop of cooling water tank level 212L200 in terms of wiring/ cabling status and electrical connection from instrument to UCP. - Check the integrity of the sensor and its installation. - If the problem persists, consider the sensor replacement at the first available opportunity

Before applying the NLP pipeline, an exploratory data analysis was conducted to better understand the peculiarities of the textual data. Among others, the analyses worth citing are words frequency and document length.

The frequency of each word in the entire corpus gives information on the used vocabulary, which is clearly characterized by its technical domain since it predominantly includes engineering words, tags, abbreviations, acronyms, machine codes, measurement values with their respective units, etc. The words frequency analysis was used to build the vocabulary, identify the main impactful acronyms, and the domain stopwords to remove, such as the words with very low or very high frequency. From Figure 1, it can be noted that words like sensor, valve or unit are among the most frequent in the vocabulary and do not add any value in describing the context of the technical report.

The documents length, namely the number of words in each technical insight, has been evaluated over time for each issue category. Figure 2 shows the number of words distribution over the documents in 2013 (blue bars) and 2019 (orange bars). This analysis did not influence the purpose of this work, however it was interesting to see through the analysis of textual data the evolutions in the way of approaching the same problem. The comparison suggests a decrease of the technical report's length over time, which may indicate a progressive improvement describing the anomalies, preferring more concise sentences to verbose descriptions.

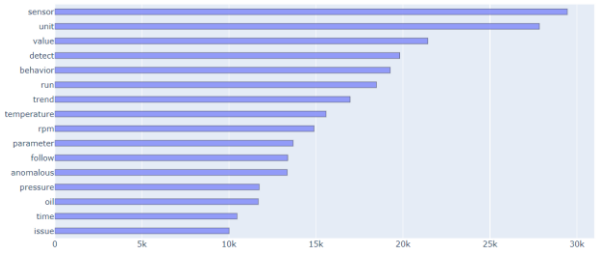

Figure 1. Words recurrency in the corpus.

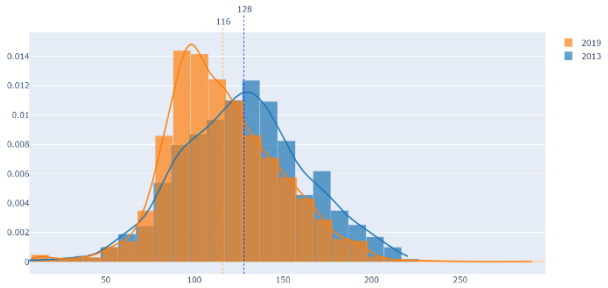

Figure 2. Documents words count in 2013 and 2019.

3. TEXT PRE-PROCESSING

Text preprocessing is typically the first step of the NLP pipeline and it may significantly impact the project outcomes (Koopman & Wilhelm, 2020; Uysal & Gunal, 2014; Kannan & Gurusamy, 2014). Often, text preprocessing is a time-consuming task and is influenced by the starting quality of data sources. Since textual data is highly unstructured, it can be noisy, can suffer bad formatting, and often, excluding language syntax, do not follow any formal rule or constraint.

Here, we describe the steps we implemented to transform the original disorganized textual data into meaningful inputs for all further processing phases. Some of these steps are very common to all NLP pipelines (like 1-2-4 and 5 below) and we mention them here only for completeness or to point out some slight customizations we needed to implement to achieve better results; the step 3 instead, is strongly connected with the technical nature of the text and therefore deserves a more in-depth description.

1. *Special characters removal*: textual data containing accents, hyphens, punctuation, multiple consecutive spaces, and empty lines have been cleaned. Removing these special characters, we increased text cleanup and made easier further analysis.
2. *Text standardization*: techniques of text standardization are fundamentals to implement good word-matching algorithms. Therefore, we converted the text to lowercase and expanded typical English contractions, thus avoiding problems when dealing with apostrophes or words outside the vocabulary. For instance, "don't" becomes "do not" removing the apostrophe and the out-of-vocabulary token "t".
3. *Acronyms, synonyms, and typos management*: a dedicated effort was made to manage the acronyms used in the vocabulary because, depending on the context, it is preferable to use the acronym or the words that generated it. To decide whether to keep the collapsed version or use the expanded one, we associated a data structure to each acronym, like the one reported in Figure 3, where the acronym "lpt" is kept in the vocabulary instead of its expanded form "low pressure turbine". Typically, acronyms concerning maintainable physical items, like "lpt" = <low pressure turbine> or "hpc" = <high pressure compressor>, are left in the shortened form, while preferring long-phrase form for the remaining ones as with "AC" = <Alternating Current> or "dps" = <differential pressure>.

```
"lpt": {
      "expanded": "low pressure turbine",
      "collapsed": "lpt",
      "regex": "\\blow pressure turbine\\b",
      "expand": false
    }
```

Figure 3. Acronym management example.

Table 2. Acronyms data structure.

Field	Description
expanded	expanded acronym
collapsed	acronym
regex	regular expression to identify the text chunk regarding the acronym
expand	Boolean to force the use of the acronym or the expanded form

The same structure has been used for synonyms and typos: in the first case, we collapsed different equivalent terms to the most suitable one and in the latter, we expanded misspelled words to the lexically correct ones. This mechanism allowed us to reduce the size of vocabulary and the noise generated by misspelled tokens and, due to the high technical content, it has been

possible only with the essential contribution of domain experts.

4. *Tokenization and lemmatization*: for tokenization task, which implies breaking the corpus in a list of meaningful elements like words, we tested several tokenizers (for instance "ToktokTokenizer" and "word_tokenizer" from NLTK[1] library) deciding at the end to use the tokenizer from spaCy[2] library, mainly for the possibility to easily customize the tokenization process on technical text.

   For the lemmatization process, which aim to replace a given token with its root, collapsing different forms of the same word to a single lemma, we adopted a custom version of spaCy lemmatizer, which, on technical terms, performs better than the standard version. For instance, the standard SpaCy lemmatizer when applied to the sentence "During the event the tumas was disabled", produces the following lemmas: 'during', 'the', 'event', 'the', 'tuma', 'be', 'disabled', erroneously considering "tumas" as a plural term (and so taking the lemma "tuma") despite the singular form of verb 'was'[3]. Our custom lemmatizer, simply filters out these tokens and treats them in the correct way, bypassing the standard lemmatizer.

5. *Standard and custom stopwords removal*: in NLP, the term stopwords refers to the words less relevant to the analysis (e.g. pronouns, conjunction, articles, etc.), which are also very frequent. In addition to the stopwords provided by NLP frameworks such as SpaCy, we created a list of about 6,500 "custom stopwords", by analyzing the lemmatized tokens with the contribution of domain experts. In this process, all the identification codes connected to physical items or to monitored signals (indecipherable tokens like "GSOV1ENAB" or "XY-113") were treated as stopwords, therefore notably increasing the size of the "custom stopwords" set.

   Both standard and custom stopwords have been removed from corpus.

As a result of this first step of the NLP pipeline, the vocabulary consisted of approximately 2,500 tokens, validated manually after domain expert's revision.

4.1 Sentence Splitting

As already stated, technical reports sent by M&D to site operators typically includes a number of recommendations to correct the detected anomalies or lower the risk. Since one goal of this project is to group similar recommendations, it was first necessary to break down the text field into individual sentences.

We firstly tried the *SpaCy* dependency parser, which is pretrained on a large number of data to build the syntactic tree of the text and identify the HEAD words at the sentence beginning. Probably, due to the technical nature of the documents, the parser performance was not satisfactory for the scope. The sentence splitting is an interesting topic in literature (Niklaus et al., 2019; Yinuo et al., 2020); we implemented a simple but effective approach based on syntactic rules (regular expressions). These rules allow to identify the end of a sentence and the beginning of the next one not only using the punctuation, but also other textual patterns typical of the domain, since full stops, colons or appropriate casing after punctuation is not always enough to decide where to split a sentence. Figure 4 shows three examples of sentence splitting; each split point is indicated by the red bar. In example 1 the colon splits the sentences while in the second one is used for listing. Moreover, points, dashes and other symbols are also used in domain entities such as sensor tags, as shown in example 3.

Example 1
RM&D suggests to: |check and verify the acquisition loop.

Example 2
Check sensor:
TT-2343-FFF
TE-DDDDD/A/B
CC-CONTROLNME

Example 3
And TT.AA176ddd wiring status of the sensor TT-WS1FI-1 [1TE6A18]. - |replace the probe if necessary.

Figure 4. Samples for sentences splitting.

4.2 Semantic Analysis - NER

Semantic analysis deals with techniques and algorithms that try to understand meaning and context of pieces of text. Named Entity Recognition (NER) is one of these techniques and concerns with the task of assigning a category to terms (or sequence of terms) that represent real-world objects.

In this project, the use of NER techniques was tested with the aim of improving the performance of the text clustering algorithms, taking into account also NER semantics (Tru et al., 2012).

We implemented a "standard NER" for categories like person, location, organizations, etc. using the spaCy EntityRecognizer and a "domain-specific NER" that recognizes peculiar entities of the project. For the latter, we made a preliminary attempt by implementing the following process:

1. use of exact matching and regular expressions to identify some domain entities defined by OEM. For instance, in the sentence "*An anomalous behavior was observed on cooling water tank level 212L200.*", the token "212L200" has been marked as a domain entity representing a water level sensor.

[1] free open-source library - https://www.nltk.org/
[2] Free open-source library - https://spacy.io/
[3] This issue occurs with the version 2.0.16 of SpaCy, but not (for example) with the version 2.3.2 where the problem has been fixed.

2. use of the found entities to create a labeled dataset to train a custom SpaCy NER model.
3. use of the developed model to extract additional domain entities.

A custom NER model was implemented using its default configurations from the SpaCy framework[4] on a dataset of about 58,000 rows (80% train set – 20% test set) generated with the approach described above. The model achieved 98% accuracy on training set and an 86% accuracy on test set, showing a lack of generalization capability. Obviously, these results needed further investigations, but not being the main objective of this work, we decided to postpone this activity and not to use the custom NER model in the final NLP pipeline.

4. LINGUISTIC MODELS

To feed machine learning algorithms with textual data, it's required to extract meaningful numerical features using linguistic models. In this work we tested three different models: bag-of-n-grams, TF-IDF and word embeddings, specifically W2V and BERT.

5.1 TF-IDF Feature Engineering

This technique has been included due its popularity and intuitiveness (Robertson, 2004) and is based on the definition of a vector space model $V = \{D_1, D_2, \cdots, D_m\}$, where m is the number of documents in the corpus and each document D_i represent the i^{th} technical case. Each document D_i is represented by a n-dimensional vector $D_i = \{w_{it_1}, w_{it_2}, \cdots, w_{it_n}\}$, where n is the size of the vocabulary and w_{it_j} is the weight of j^{th} term in i^{th} document.

To implement TF-IDF feature engineering, we used the class "TfidfVectorizer" from the open source *scikit-learn* Python library, where the weights are defined as follow:

$$w_{it_j} = tf(t_j, D_i) * idf(t_j)$$

with:

$tf(t_j, D_i)$ = number of occurrences of t_j in D_i

$$idf(t_j) = 1 + \log\frac{1+m}{1+df(t_j)}$$

$df(t_j)$ = number of documents containing the term t_j

The weighting schema above has been applied to the vocabulary created with the following parameters:

1. $ngram\_range = (1,2)$ i.e. the vocabulary is made up of single terms and bigrams to preserve minimal information about words order.
2. $min\_df = 3$ i.e. the vocabulary contains unigrams and bigrams present in at least three documents, ignoring in this way very rare terms.
3. $max\_df = 0.5$ i.e. unigrams and bigrams present in more than half of documents are not included in the vocabulary, discarding very common terms like 'sensor' and 'detect' or bi-grams like 'run follow'.
4. $max\_features = 10{,}000$ i.e. only unigrams and bigrams with frequency ranked in top 10,000 positions are retained, considering this value a good threshold for our corpus.

5.2 Word Embeddings

In addition to bag-of-n-grams model, based on the words frequency in the corpus without taking into account semantic or disambiguation, embedding modeling has also been used as feature extraction approach.

Word embeddings are a numerical representation of the meaning of words, where words are mapped to vectors of real numbers using a set of language modeling techniques. These techniques typically involve Neural Networks (NN) that are trained starting from an un-annotated corpus, capable of memorizing semantic information by constructing a n-dimensional vector space in which the word vectors are close if the words occur in the same linguistic contexts.

For instance, Figure 5 shows the existing interconnections between the term "oil", which refers to the lubricating system of turbomachinery, and other related terms such as "lube", "tank", "mineral" and "synthetic". In the text corpus, such words appear jointly or in a nearby position within the sentence. To plot the n-d space data into a 2-d space has been used the t-SNE (van der Maaten & Hinton, 2008) dimensionality reduction technique.

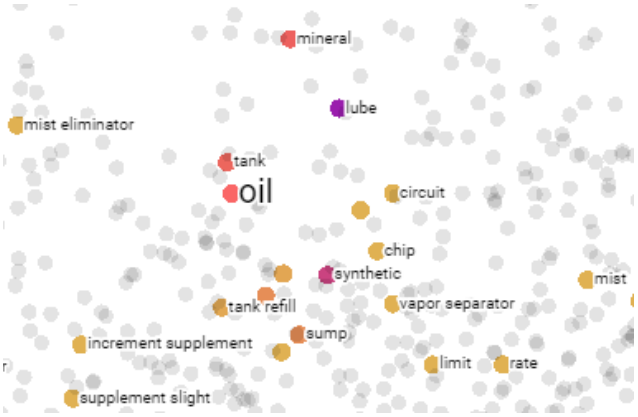

Figure 5. Words connected with the term "oil".

[4] for details, see https://v2.spacy.io/usage/training#ner

Two families of algorithms were used, *word2vec* (Mikolov et al, 2013) and *BERT* (Devlin et al, 2018) that belongs to the *transformers* (Vaswani et al, 2017) family.

5.3 Word2Vec

For the training of the Word2Vec model, the skip-gram algorithm has been selected, which predicts the context (or neighbors) of a word given the word itself.

The context of a word can be represented through a set of skip-gram pairs of (*target_word*, *context_word*) where *context_word* appears in the neighboring context of *target_word*. The context words are defined by a window size that determines the span of words on either side of a target_word that can be considered context word.

The training objective is to maximize the probability of predicting context words given the target word. For a sequence of words $w_1, w_2, \cdots, w_t$ the objective is the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

where c is the size of the training context. The basic skip-gram formulation defines this probability using the *softmax* function:

$$p(w_O|w_I) = \frac{exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{W} exp(v'^T_{w_O} v_{w_I})}$$

where v and v' are respectively target and context vector representations of words and W is the vocabulary size.

Using the *gensim*[5] library, a set of blank models have been trained on the dataset with the following parameters:

1. size: [100, 300, 768, 1024] - the size of the embeddings.
2. window: [5, 30] - the dimension of the context in which the predicted words are present starting from the target word.
3. alpha: 0.025 – the learning rate.

Each model was able to return an embedding for each vocabulary's word. Finally, for each document of the dataset has been calculated a mean vector by averaging the word embeddings of the words in the documents. Figure 6**Erreur ! Source du renvoi introuvable.** shows the distribution of a subset of the recommendation dataset in a 2-d space, where each document is embedded in a 100-dimensional space. The dimensionality reduction was performed by using the t-SNE algorithm. For the specific case shown in the figure, it can be noted the clear presence of some clusters.

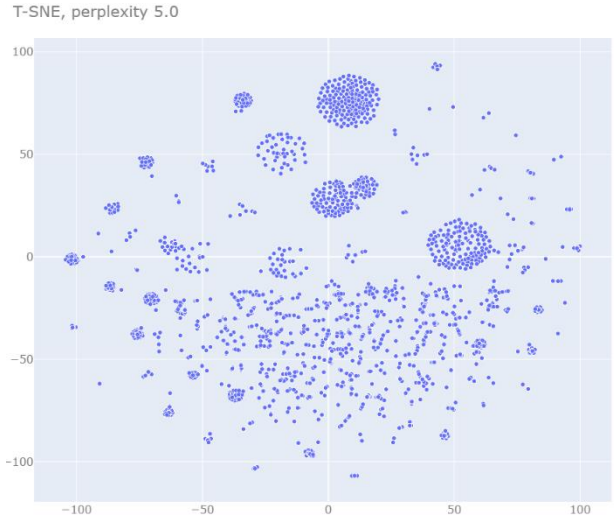

Figure 6. Distribution of a subset of the recommendation dataset (bearing anomalies) by word2vec modeling.

5.4 BERT

Another suitable model for NLP tasks has been tested as well (Ethayarajh, 2019). *Bidirectional Encoder Representation from Transformers* (BERT) is one of the state-of-the-art language models for a wide variety of NLP tasks (Devlin J. et al, 2018). It makes use of *transformers* architecture based on an *attention* mechanism (Vaswani et al., 2017) to learn contextual relations between words in a text by an encoder-decoder sequence. The encoder reads the input text that is first embedded into vectors and then processed in a deep Neural Network (NN) and the decoder produces a prediction for the task. Since the purpose is to generate a language model, only the encoder part is necessary.

BERT's key technical innovation is applying the bidirectional training of transformers to language modeling that encodes the entire sequence of words at once allowing the model to contextualize a word based on all its surroundings words, this makes BERT context dependent and it is shown that this approach works really well for many NLP tasks (Peters M. E. et al, 2018). Bidirectional training can have deeper sense of language context and flow than a single-direction language model (Devlin J. et al, 2018).

For this work has been used the basic pre-trained model, provided by the *huggingFace*[6] library, then a *transfer learning* has been performed by appending one hidden layer to the core model and a decoder at the head of the NN. The

[5] Free open-source library https://radimrehurek.com/gensim/

[6] Free open-source library https://huggingface.co/

basic configuration of this model counts more that 108M parameters, with 12 hidden layers and an embedding size of 768, and it is pre-trained on an enormous dataset of texts (Wikipedia content + BookCorpus[7]). As in the original paper (Devlin J. et al, 2018), the appended layer has been trained with a Masked Language Model (MLM), in which 15% of the words in each sequence was replaced with a token, in particular the 80% of the times with a [MASK] token, the 10% with a random token and the remaining 10% was not replaced at all. In this way, the model attempts to predict the original value of the masked words based on the context provided by other non-masked words. The input documents have been padded to fix the input size then trained for several epochs with a step decay schedule to drop the learning rate by a factor $\gamma = 0.8$ every 20 epochs.

The objective was to minimize the loss function of the decoder at the head of the NN, the decoder maps the 768-dimension space of the embedded into a 30522-dimension space vector, where 30522 is the vocabulary size available from basic BERT model and represents the probability for each element in the vocabulary to be predicted. We trained several models with following parameters:

1. epochs: [50, 100 ,200, 250, 314]
2. starting learning rate: 0.05
3. step LR: $\gamma = 0.8$, every 20 epochs learning rate update as follows:

$$lr = lr * \gamma$$

After the model training of both word2vec and BERT, the documents vector representations were obtained by using only the encoder part of the model. These vectors were then tested for the clustering task.

5. TOPIC MODELING AND TEXT CLUSTERING

Topic modeling and text clustering represent the most widely used methods to classify and organize documents in a corpus. The purpose of topic modeling is to discover distinguishable concepts embedded in a set of documents, assuming a probabilistic hypothesis according to which similar terms occur in similar contexts.

On the other hand, text clustering implies the definition of a similarity measure (or equivalently a distance measure) between the feature vectors representing documents, with the purpose of partitioning them into coherent groups (eventually discarding documents considered as noise).

In this project, the implementation of these two methods turned out to be very challenging due to the following factors:

1. the training process is completely unsupervised, meaning that all the documents in the corpus are unlabeled and we had no previous clue about the final number of involved clusters/topics.
2. the application of clustering algorithms to textual data is often challenging due to many factors (Afzali & Kumar, 2019) and the boundary between obtained clusters is rarely high defined. A domain-expert effort dedicated to the analysis and inspection of results is therefore required.
3. as consequence of the previous point, the common metrics used to evaluate the performance of clustering methods (e.g. elbow method, silhouette method, etc.) do not give a clear indication about the optimal number of clusters in the dataset. Also in this case, the domain expertise is crucial.

6.1 Topic modeling

Topic modeling is a common method used to analyze unclassified textual data to group together features (unigrams and bigrams in this project) with similar meaning, in order to identify distinguishable concepts or themes. The extracted topics can give a glimpse of the subjects covered by the corpus and, at the same time, can create semantic connections between words in the same topic.

To determine the optimal number of topics, Latent Dirichlet Allocation (Blei et al., 2003) method of "gensim" library was used, and the best result was selected after several tests by varying the model hyperparameters. The model performance assessment was done using the following metrics:

1. "Perplexity" to measure how well the LDA model describes the actual distribution of words in documents (lower perplexity implies a better model).
2. "UMass" and "c_v" to measure topic coherence (lower "UMass" and higher "c_v" implies a better model).

whereas the tuning hyperparameters are the number of topics, the document-topic probability (α), and the word-topic probability (β).

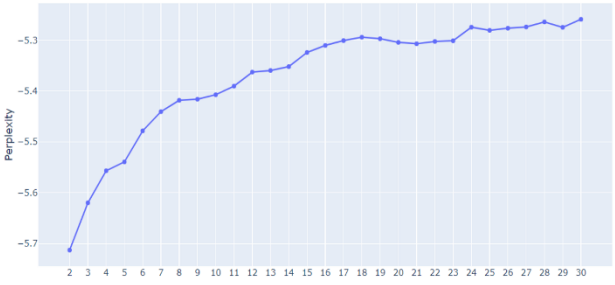

Figure 7. Perplexity metric by varying the number of topics.

[7] Free open-source data https://huggingface.co/datasets/bookcorpus

For example, setting $\alpha = 0.001$ and $\beta =$ 'auto' (meaning that the hyperparameter is learned from data) we have the perplexity values shown in Figure 7 and the coherence values shown in Figure 8. From the analysis of results, we selected the number of topics $K = 17$; the results in terms of topics and words, are shown in Figure 9. The outcome has been manually validated by inspecting the words occurrence within the topics, and by studying the effect of changing the value of K on the overall model performance.

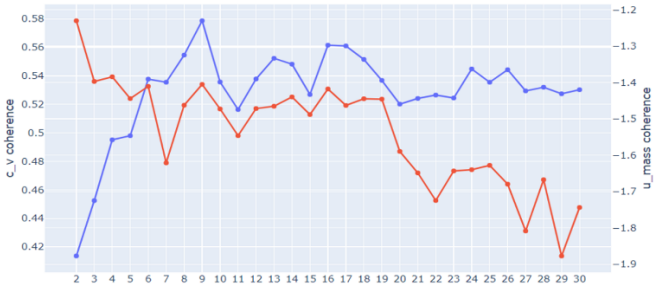

Figure 8. Coherence metrics by varying the number of topics (blue is "c_v", red is "UMass")

In Figure 9, generated using pyLDAvis[8] (Sievert & Shirley, 2014), after selecting a specific topic on the left side of the graph, represented by a numbered circle, the 30 most relevant words of the selected cluster are visualized on the right side. Moreover, each word is represented by a bicolored bar containing two types of information: the (estimated) frequency of the word within the selected topic (red) and the overall word frequency (blue) in the corpus.

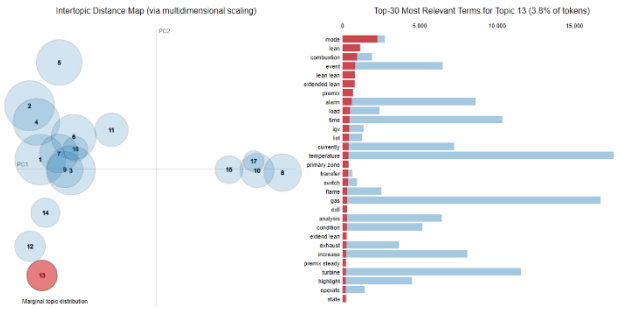

Figure 9. Topics and most relevant words per topic.

The overlapping bubbles in Figure 9 revealed the difficulty to distinguish between the topics involved and this mainly comes from the common tech lexicon used for different areas of intervention.

6.2 Document Clustering

An alternative approach to identify subjects and topics existing in a corpus is to define a distance between documents and then use clustering algorithms to partition data into coherent groups.

For each of the above-mentioned linguistic models, we applied the following clustering algorithms:

1. *K-Means:* a partitioning algorithm based on Euclidean distance that tries to minimize the within-cluster sum-of-squares (MacQueen, 1967).
2. *Spectral clustering:* an algorithm that first makes a dimensionality reduction using similarity matrix and then applies K-Means on a fewer-dimension space (Jianbo and Jitendra, 2000 – Ng et al., 2002).
3. *Hierarchical clustering*: starts from m clusters (one for each document) and then merges pairs of clusters using a measure of (dis)similarity (Murtagh & Contreras 2011).
4. *Affinity propagation:* does not require in advance the number of clusters (Frey & Dueck, 2007) and is based on message passing between data point: each data point is seen as a node of a network and the clustering structure emerges as consequence of a recursive transmission of messages along the edges of the network. In particular, using the initial set of similarity values, two kind of message are exchanged between nodes: (a) "responsibility" $r(i,k)$, from data points to candidate exemplars, indicating how strong node i prefers node k as cluster exemplar over other nodes (b), and "availability" $a(i,k)$, from candidate exemplars to data points, indicating the availability degree of exemplar k of being cluster center for the data point i.
5. *HDBSCAN:* a density-based algorithm (Campello et al., 2013) that, unlike the other methods, is not a partitioning algorithm hence can exclude documents from final clusters, considering them as noise.

To evaluate the clustering validity in terms of tightness/separation and (when needed) to support the selection of the appropriate number of clusters, the main adopted metric was the *average silhouette* (Rousseeuw, 1987) together with other metrics peculiar to the algorithm under evaluation (e.g. SSE = sum of square errors for K-Means).

All combinations of linguistic models and clustering algorithms were extensively tested. For linguistic *models* the best results were achieved with TF-IDF and W2V compared to pre-trained BERT. Regarding clustering *algorithms,* "K-Means", "Spectral" and "Hierarchical" achieved good results, "Affinity propagation" showed lower performance and problems on speed execution and memory occupation, while on "HDBSCAN" we encountered issues in tuning the hyperparameters.

[8]Open-source library https://pyldavis.readthedocs.io/en/latest/readme.html

As an example of eligible clustering partitioning, we can consider the spectral clustering algorithm applied to TF-IDF model. The average silhouette metric is shown in Figure 10 and the value of $K = 42$ ($sil = 0.16$) could be selected as an acceptable number of clusters to consider. The clustering partitioning resulting from this model is shown in the 2D dimensional space of Figure 11.

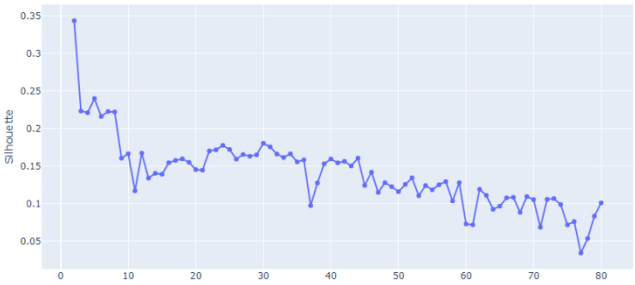

Figure 10. Average silhouette versus number of clusters.

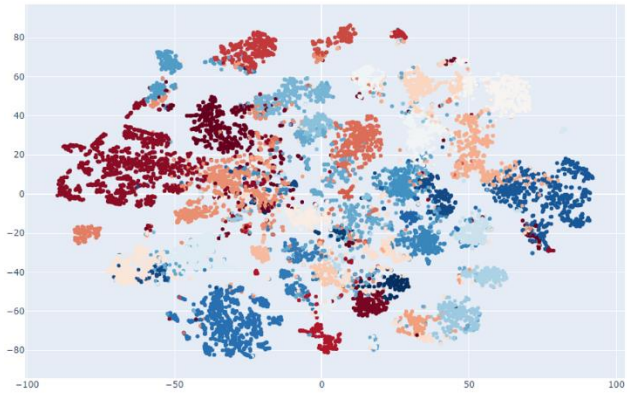

Figure 11. 2D view of spectral clustering results for K=42.

The clusters obtained with K=42 were manually checked and labeled. Some examples are shown below:

a. *Oil Consumption*: containing keywords like 'synthetic' – 'oil' – 'tank' – 'synthetic oil' – 'consumption' - 'rate'– 'oil tank' – 'synthetic lube' – 'lube' – 'lube oil'

b. *Compressor axial displacement*: containing keywords like 'displacement' – 'axial displacement' – 'axial' – 'compressor' – 'shaft axial' – 'shaft' – 'thrust' – 'speed' – 'displacement erratic' – 'compressor axial'.

c. *Air filter*: containing keywords like 'filter' – 'air' – 'differential' – 'differential pressure' – 'inlet' – 'pressure' – 'filter differential' – 'inlet filter' – 'air filter' – 'air inlet'.

revealing three different areas of intervention reported by M&D on corresponding problems.

The procedures and outputs shown so far, using several language models and clustering algorithms, allowed us to aggregate and label similar M&D technical reports, with a particular focus for the dataset of description and technical assessment.

The same process has been applied also to the dataset of recommendations, i.e. the actionable insights to site operators contained in the M&D technical reports. Since 2013, roughly 25000 recommendations have been identified; among those, some were identical, some others were giving similar indications using different wording, only a smaller percentage contained a unique content. Applying the above work process (text modeling and clustering) on this data, we found about 850 different clusters, each of them representing a unique relevant recommendation.

The recommendations' clusters lay the foundation for creating a library of standard recommendations: these results are currently being analyzed by domain technical experts for this purpose. Moreover, these outputs have been implemented in the web app for testing and validation purposes.

6. WEB APP

Leveraging these outputs, a prototype of web application was then developed to support technical operators during the creation of new insights for customers, from the drafting of the technical assessment to the suggestion of the most relevant recommendations for the problem being treated.

The web app has been developed using an internal framework based on Flask, which allows the fast prototyping of analytics and web applications.

For the first release of the web app, we decided to use TF-IDF and k-means clustering algorithm which are simpler and more interpretable than the word-embeddings models (Haitong et al., 2020).

The web application makes available to end users a set of features, described below:

- *cluster identification* - given a user defined text input, such as a failure mode or a technical case description, it is possible to obtain the most relevant cluster, together with its main keywords, calculated as the most frequent words of that cluster, and the top 10 technical reports, ordered by centroid distance.
- *similar cases search* - given a user defined text input, such as a technical case description, the app retrieves the top 10 similar cases from the database, ordered by similarity score where similarity is calculated using cosine distance.
- *recommendations suggestion* - given a user defined text input, such as a technical case description, the tool suggests a set of relevant recommendations retrieved as follow: using the input text, *cluster identification* feature is invoked, and the most relevant cluster, along with its technical reports, are retrieved. All the recommendations

contained are then processed and mapped with the corresponding clusters, sorted by a coefficient proportional to the occurrence in the dataset, and finally suggested to the end user.

- *trending topic* – this feature allows to display all the technical reports' clusters in a 2-d plot where each cluster size is proportional to the number of notifications released in a period of time selected by the user.

In Figure 12 is shown an example of cluster identification, where the input string "lube oil consumption", which refers to an issue of the gas turbine oil system, returns the cluster no. 32 as the most likely, together with its keywords and two technical cases leading to it.

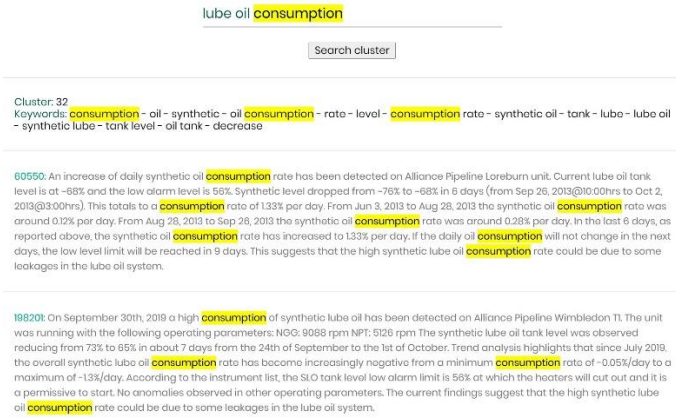

Figure 12. Web app retrieving cluster given an input string.

Currently, the web application is being tested and feedbacks from end users are being collected to better understand how and where to improve the tool and the models in the backend. Tests using different combinations of linguistic models and clustering algorithms will follow in future releases.

7. CONCLUSIONS

This work is a first attempt to exploit the OEM knowledge contained in the textual documents of the technical maintenance reports that Baker Hughes regularly sends to site operators. The paper describes the nature of the textual documents analyzed, their peculiarities, all the preprocessing steps performed, and the NLP models tested.

Several linguistic models and clustering techniques have been tested on the available text data and a first prototype of web app has been made available to operators for testing purposes.

The capabilities giving a benefit to the M&D service, for which the web app represents a first draft, are the following:

- retrieve a technical report draft, using the most relevant cluster in addition to search tools already implemented (like Solr / Elastic Search).
- suggest relevant recommendations, based on the content of a technical report (event description and technical assessment) already prepared by the subject matter expert.
- look for similar technical reports, to see how similar issues were managed and find out if different considerations were made by experts.

Additionally, information retrieval and NLP could serve to build structured feedbacks to improve the technical support, e.g. capturing the feedback from site and closing the loop, or to improve the machine engineering design.

Next steps for the presented work, include further tests on the BERT model, in particular the evaluation of different implementations and a better tuning of model hyperparameters, starting from the iterations number and the number of features.

A supervised approach can also be considered; since the labelling effort on historical data can be considerable, a semi-automatic labeling step can be embedded in the technical support work process, thus obtaining a good amount of labeled documents with a low distributed effort.

ACKNOWLEDGEMENTS

We would like to thank Marco Rajola and Domenico Iasparro, for their contributions in use cases and guidance on Web App development.

REFERENCES

Afzali, M., Kumar, S. (2019). Text Document Clustering: Issues and Challenges. *263-268. 10.1109/COMITCon.2019.8862247.*

Blei D., Ng A., Jordan M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Brendan J. Frey and Delbert Dueck (2007). Clustering by Passing Messages Between Data Points, *Science*.

Brundage M.P., Sexton T., Hodkiewicz M., Dima A., Lukens S., (2020). Technical language processing: Unlocking maintenance knowledge*, Manufacturing Letters, Volume 27, January 2021, Pages 42-46*.

Campello Ricardo J. G. B.,Moulavi Davoud, Sander Joerg (2013). Density-based clustering based on hierarchical density estimates. *Advances in Knowledge Discovery and Data Mining*, Springer, pp. 160-172.

Devlin J., Chang M, Lee M.-K., Toutanova K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Ethayarajh K. (2019). How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. arXiv: 1909.00512.

Kannan, S., Gurusamy, V. (2014) Preprocessing Techniques for Text Mining. *IJCSC, 5, 7–16*.

Koopman, C., Wilhelm, A. (2020). The Effect of Preprocessing on Short Document Clustering. *6. 10.5445/KSP/1000098011/01*.

Haitong Z., Yongping D., Jiaxin S., Qingxiao L. (2020). Improving Interpretability of Word Embeddings by Generating Definition and Usage. *Expert Systems with Applications, Volume 160, 1 December 2020, 113633, arXiv:1912.05898.*

Iannitelli M., Vairo V., Parrella I., Pedullà G. (2018). A digitalized approach for combining diagnostic capabilities and maintenance risk-based insights to improve machine operation. *The Future of Gas Turbine Technology 9th International Gas Turbine Conference*, October 10-11, Brussels, Belgium.

Jianbo S. and Jitendra M. (2000), Normalized Cuts and Image Segmentation. *IEEE Transactions on PAMI, Vol. 22, No. 8*.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press. pp. 281–297.*

Mikolov T., Chen K., Corrado G., Dean J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.

Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pages 3111–3119.

Murtagh F. and Contreras P. (2011). Methods of hierarchical clustering. *arXiv:1105.0121*.

Ng, A., Jordan M., Weiss Y. (2002). On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems*.

Niklaus C., Cetto M., Freitas A., Handschuh S. (2019). *Transforming Complex Sentences into a Semantic Hierarchy.* arXiv:1906.01038v1.

Peter J. Rousseeuw (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics*, 20: 53-65.

Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L. (2018). Deep contextualized word representations. arXiv:1802.05365.

Robertson S (2004) Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation 60*.

Sexton T., Hodkiewicz M., Brundage M.P., Smoker T. (2018). Benchmarking for keyword extraction methodologies in maintenance work orders. *PHM Society Conference; vol. 10*.

Sexton T., Hodkiewicz M., Brundage M.P. (2019). Categorization errors for data entry in maintenance work-orders. *Proceedings of the Annual Conference of the PHM Society; vol. 11*.

Sharp M., Brundage M., Sexton T. and Madhusudanan F. (2021). Discovering Critical KPI Factors from Natural Language in Maintenance Work Orders, *ASME Journal of Manufacturing Science and Engineering*.

Sievert C. & Shirley K. E. (2014). LDAvis: A Method for Visualizing and Interpreting Topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces 63–70*.

Stenström C., Al-Jumaili M., Parida A. (2015). Natural language processing of maintenance records data, *International Journal of COMADEM, ISSN 1363-7681, Vol. 18, no 2, p. 33-37.*

Tru H. Cao, Thao M. Tang, Cuong K. Chau (2012). Text Clustering with Named Entities: A Model, Experimentation and Realization. *Book Chapter in Intelligent Systems Reference Library, 1, Volume 23, Data Mining: Foundations and Intelligent Paradigms - Springer, pp. 267-287*.

Uysal, A. K., Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management. 50. 104 - 112. 10.1016/j.ipm.2013.08.006*.

Van der Maaten L.J.P., Hinton G.E. (2008). Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9:2579-2605.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. (2017). Attention is all you need, arXiv:1706.03762.

Yinuo Guo, Tao Ge, Furu Wei (2020). *Fact-aware Sentence Split and Rephrase with Permutation Invariant Training*. arXiv:2001.11383v2.

BIOGRAPHIES

Daniele Pau is a Data Scientist at Baker Hughes, Firenze, Italy. He received his master's degree in biomedical engineering from University of Pisa in 2018. In 2020 he earned a postgraduate specialization in Big Data Analytics & Social Mining from University of Pisa. His experience includes knowledge in software developing and machine learning, especially in the field of Natural Language Processing. He currently works in the Commercial and Global Service, involved in customer profiling project and maintenance policy planning platform.

Isaia Tarquini is a data scientist and experienced software architect/developer. He received his master's degree in computer science and his postgraduate master in Big Data Analytics & Social Mining from University of Pisa. In his current role, Isaia is working in the field of Natural Language Processing and Information Retrieval specifically in recommendation and expert finding systems.

Matteo Iannitelli is a Lead data scientist at Baker Hughes, Firenze, Italy. He received his master's degree in automation engineering from University of Pisa. He currently works in Monitoring & Diagnostic department developing and implementing AI algorithms for predictive maintenance and failure identification. He is also leading the functional development of an internal big data analytics platform.

Carmine Allegorico is a Principal engineer and experienced data scientist at Baker Hughes, Firenze, Italy. He received his master's degree in mechanical engineering from University of Napoli Federico II. In his current role, Carmine is a technical point of reference for the analytics discipline providing engineering guidance to other teams, helping to train new engineers and keeping abreast of industry trends and issues. He provides consulting during the development and implementation of advanced solutions for the on-line diagnostic and predictive maintenance, coordinates the creation of internal processes and support the adoption of new platforms and technologies.